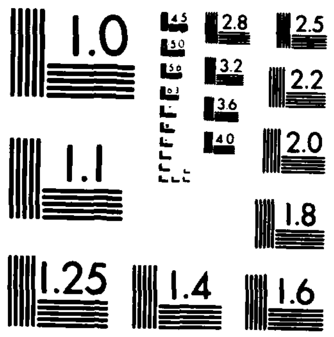
AD-A150 423 ALIAS (ACQUISITION AND LOGISTICS INFORMATION AND ANALYSIS SYSTEM) MAINTEN..(U) DECISION-SCIENCE APPLICATIONS INC ARLINGTON VA M S CAREY ET AL. 1/7

UNCLASSIFIED 31 OCT 84 DSA-593-VOL-2 N00014-82-C-0813 F/G 15/5 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

13

DSA DECISION·SCIENCE APPLICATIONS

DSA Report #593

October 31, 1984

ALIAS MAINTENANCE AND EXPANSION GUIDE

VOLUME II

Submitted to:

Scientific Officer
Naval Center for Acquisition Research
NAVMAT 08
Washington, D.C. 20360

Attention: Dr. Thomas C. Varley

DSA ***DECISION-SCIENCE APPLICATIONS***

DSA Report #593 October 31, 1984

ALIAS MAINTENANCE AND EXPANSION GUIDE
VOLUME II

M.S. CAREY
J.C. KRUPP
J.M. KABAT

Submitted to:

Scientific Officer
Naval Center for Acquisition Research
NAVMAT 08
Washington, D.C. 20360

Attention: Dr. Thomas C. Varley

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Direct (Naval Sea Systems Command(SEAOOD)) Washington, D.C. 20362
4. PERFORMING ORGANIZATION REPORT NUMBER(S) DSA Report No. 593	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Decision-Science Applications, Inc. 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Navy Office for Acquisition Research
6c. ADDRESS (City, State and ZIP Code) 1901 North Moore Street, Suite 1000 Arlington, Virginia 22209	7b. ADDRESS (City, State and ZIP Code) NAVMAT 08 Washington D.C. 20362
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Office of Naval Research 8b. OFFICE SYMBOL (If applicable) ONR	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-82-C-0813
8c. ADDRESS (City, State and ZIP Code) Department of the Navy 800 North Quincy Street Arlington, Virginia 22217	10. SOURCE OF FUNDING NOS.

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
N68342	980770		

11. TITLE (Include Security Classification)
ALIAS Maintenance and Expansion Guide, Volumes I and II

12. PERSONAL AUTHOR(S)
M. Carey, J. Krupp, J. Kabat

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Technical	FROM 10/82 TO 8/84	31 October 1984	

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
ALIAS, DATA BASE, DBMS, MODELS, STRUCTURED PROGRAMMING

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This documentation explains the structure of the Acquisition and Logistics Information and Analysis System (ALIAS). With this documentation, the experienced programmer should be able to easily maintain and expand the ALIAS system. In addition, the manuals explain all standards to which ALIAS extensions should conform. For the non-programmer these manuals describe the philosophy of ALIAS and its extent and limitations.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☒ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Joseph C. Krupp	703-243-2500	DSA

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE. UNCLASSIFIED

TABLE OF CONTENTS
VOLUME II

TABLE OF CONTENTS

VOLUME II (Continued)

LIST OF TABLES

VOLUME II

LIST OF FIGURES
VOLUME II

10.0 ALIAS UTILITIES AND COMMON DATA STRUCTURES

Within the ALIAS system, a utility is a FORTRAN subroutine or BUILDER screen which performs a well-defined, limited task in such a way that it can be used by many system processors. For example, the CCAT2 subroutine is used by virtually all ALIAS FORTRAN programs when they want to concatenate two character strings.

A routine is typically not considered a utility if the function it performs is of interest to only one program, or if it requires that a complex global data structure be in place if it is to work properly. The main motivation for using utilities is that they save the programmer time; if the programmer must go to a lot of trouble getting set up to use a given "utility", he is less likely to find it a time-saver.

That said, some ALIAS utilities do require that a global data structure be in place if they are to work properly, but this structure is almost always: (1) the System Core data structure, which is always in place during an ALIAS run, or (2) a structure that can be initialized by a call to single initialization routine.

This Section will list and describe ALIAS utilities. It is meant to be a reference for programmers engaged in development work.

The Section will also present all ALIAS FORTRAN include files (which typically contain common block definitions); these are like utilities in that they are global resources often used by more than one processor.

Miscellaneous system resources, such as extra data segments, will also be covered.

The theme for Section 10, then, is coverage of shared system resources: anything used by more than one ALIAS module or by more than one part of the Core will be covered here (but not the data base---see the ALIAS Data Base Reference Manual).

ALIAS FORTRAN utility routines fall into three categories:

1) Linkable general-purpose routines, where "linkable" means their object code is included in a program at PREP time.

2) Data Base management system InterFace routines (DBIF). These buffer requests for DBMS services. They are more programmer-friendly than RELATE HLI routines, and make ALIAS more convertible by isolating the calls that depend on the particular DBMS being used.

3) BUILDER-callable routines. Residing in the account Segmented Library, these routines are designed to serve BUILDER screens (via the BUILDER CALL PROCEDURE command). They are linked at or after RUN time. It is possible to call some of them from normal FORTRAN programs, but this is not usually advisable.

The next three subsections will discuss each class of FORTRAN utility in turn. Section 10.4 will discuss BUILDER utility screens. Section 10.5 presents ALIAS FORTRAN include files, and Section 10.6 discusses miscellaneous global resources.

10.1 GENERAL PURPOSE FORTRAN ROUTINES

ALIAS general-purpose FORTRAN utilities reside in the UTLO, UTLR, and RECOMP libraries. Source code is in utlo.src, recomp.src, and utlr___.src (there are several utlr source files; routines appear in them in alphabetical order according to the usual naming conventions). Object code is in utlo.obj and recomp.obj (normal object code files whose contents must be copied into an object code file about to be PREPed), and in utlr.obj. Utlr.obj is a Relocatable Library (RL), a special HP file which can be specified as a place for PREP to look for unsatisfied externals. This file should never be specified as the target for compilation; object code must be moved into this file from a regular object code file by explicit SEGMENTER commands.

Note that any routine <u>called</u> by a routine in an RL must also be in the RL (or else in the SL). This is why UTLO is maintained in addition to UTLR. UTLO contains utilities which are inconvenient to keep in the RL, typically because they use common blocks which change occasionally. It is tedious to have to do the recompilation and then replace the object code copy in the RL as well.

If utilities are self-contained it is more convenient to keep them in the RL because the amount of SEGMENTER work necessary to build up an object code file suitable for processing by PREP is reduced.

This Section is meant as a reference to allow programmers to quickly locate utilities of use to them, or to find more information about utilities they are having difficulty with. The utilities will be divided into about 20 groups by purpose, and a brief introduction to each group will be provided. Table 10-1 lists the groups; Table 10-2 is an annotated listing of the utilities in each group, the annotations describing the purpose of each routine. Programmers looking for a utility to perform a specific task will hopefully be able to find it quickly by consulting Tables 10-1 and 10-2.

Detailed information about any given utility will be found in Section 10.1.2, which contains the abstract/header from each utility routine in alphabetical order. These describe what arguments are required and the operation of the routine in more detail.

Programmers with the opposite problem, a specific utility which they want to know more about, can find out which group the routine belongs to and where its source code is located by referencing the alphabetical listing of Table 10-3.

Table 10-1. Types of General-Purpose FORTRAN Utility

PURPOSE	DISCUSSION
BIT MANIPLUATION	Bit comparison, bitwise and/or
CHARACTER STRING MANIPULATION	String operations, e.g. concatenation, parsing, uppercasing, etc.
DATA MOVEMENT	Transfer data from one array to another. Also array initialization.
DATA RANGE CHECKING	What kind of characters in string? Number too big?
DATA TYPE CONVERSION	ASCII to numeric and vice versa.
DATE MANIPULATION	Any date-oriented operation you can imagine.
DEVICE CONTROL	Aids for sending hard copy output.
DIAGNOSTICS	lprnt setting aids.
ERROR MESSAGE OUTPUT	Means to tell the user things are messed up.
FILE OPEN/INPUT/OUTPUT	A near-F77 OPEN and some direct access helpers
FORMATTERS	For paged output and for bulk text output.
INITIALIZATION	Never hurts to call these, often helps.
LINE INPUT	Retrieve the next input line from anywhere.
MATHEMATICS	Mainly vector operations.
MEMORY MANAGER	An interface to extra data segments.
MISCELLANEOUS	Various goodies.
OPERATING SYSTEM INTERFACE	These make the intrinsic calls for you.
SORTING SEARCHING	Find a match, sort an array.
STACK DATA TYPE	Implementation of a stack data type.
USER INTERACTION	Prompting utilities.

Table 10-2. General-Purpose FORTRAN Utilities By Type

NAME	PURPOSE
	BIT MANIPLUATION
IAND IOR	IOR is an entry point in IAND. These do AND and OR tests on the least significant bit of a 16-bit number. They are integer functions, returning a 1 or 0.
LBIT	Logical function returning true if a given bit in a 16-bit word is set.
	CHARACTER STRING MANIPULATION
CCAT2 CCAT3 CCAT4	CCAT4 is an entry point in CCAT3. These concatentate 2, 3, or 4 separate strings into a single output string. Input strings may be delimited. An input string may be specified as the output target.
CEQ	Logical function which strips trailing blanks only before performing an equality test on two strings. Use CIF in general.
CHNALO CHNDEA CHNFRE CHNINI	These support string-chain data types which manages a string buffer space, allowing efficient storage of large strings. See CHNINI abstract. Note the routines can support many buffers; the buffer data structure is not built into them.
CIF	Logical function which strips leading and trailing blanks from two input strings and then compares them. Useful since HP automatically considers two strings of different length or with different blank-padding not equal.
DELIM	Useful in extracting from a delimited string.
DINDEX	Integer function which performs an index on a delimited string.
ELIMBL	ELIMinate BLanks. Left-justifies a string and returns its non-blank length.
LOWERC	Converts all letters in a string to lower case. An entry point in UPPERC.
LSTRNG	Undelimits a string and left-justifies it in the output buffer.
LTRIM	Integer function giving the location of the leftmost non-blank character. Returns length+1 if all blanks.

Table 10-2. General-Purpose FORTRAN Utilities By Type

NAME	PURPOSE
MPDCOD	Parser which splits a delimited string further delimited internally by commas into its constituent parts, placing them in an output array. Now used only by filopn.
MSTRNG	Makes the input string into a delimited string. This is an entry point in LSTRNG.
NCFLW NCFW NCSW NLWFC NSWFC NSWFW NWFC NWFSW	All these routines are entry points in NCFW. They convert array length in number of *2, *4, and *8 words into lengths in bytes, and vice versa.
RTRIM	Integer function returning the location of the rightmost non-blank character. Returns 0 if all blanks.
UPPERC	Uppercases all letters in the string.
	DATA MOVEMENT
XMIT XMITB XMIT2 XMIT2B XMIT4 XMIT4B XMITC XMITCB	These routines are just assignment loops which transfer data from one array to another. Xmit and xmit4 do reals or integer*4, xmit2 integer*2, and xmitc characters. They can reduce the volume of code in your routines by doing the work of loops with only one line. Also, if their number-of-words-to-transfer argument is negative, they expect the source to be a single word (byte) which they are to fill the target with. They can thus be very handy for array initialization. Note xmit expects to loop a *2 number of times, xmit4 a *4 number of times, so be careful specifying arguments. Also, the regular entry points can left-shift data (i.e. move second element of an array into first element, third into second, etc.) while the "B" entry points can right-shift.
	DATA RANGE CHECKING
ASCINT	Returns true if string has ASCII integer characters only.
ASCPRN	Returns true if string contains only printing ASCII.

Table 10-2. General-Purpose FORTRAN Utilities By Type

NAME	PURPOSE
ASCREL	Same as ASCINT but permits ".", thus allowing real numbers.
BETWN	Returns true is an integer lies between specified low and high values.
CRNGI CRNGI4	Generates an abort if an integer's value lies outside a specified range. The 4 version is for *4 integers.
LETNUM	Returns true if a string contains letters and numbers only, i.e. no "%", etc.
LETONL	Returns true if a string contains letters only.
	DATA TYPE CONVERSION
FLT	Converts a string to a real number. Same as the corresponding FORTRAN intrinsic, but this routine returns an error flag instead of a system abort if it can't do the job.
KFIX	Converts a string to an integer (*2). Returns an error flag true (rather than an abort) if string cannot be converted.
NUMASK	Converts an integer number into a character string AND right-justifies it into a given character string. E.g., 234 and "00000" come out as "00234"; -12 and "0000" as "0-12". If the number is too big to fit then "****..." are output, conforming to the usual FORTRAN convention. Useful in output construction.
NUMSFX	Character function returning a labeling suffix for a number, in caps or lower case. E.g., 5 leads to output of "th" or "TH"; 1 to output of "st" or "ST". Useful in constructing custom-formatted output.
PLURAL	Similar in purpose to numsfx. Character function returning "s" or "S" if number input is 1, blank otherwise.
STRN	Like the FORTRAN intrinsic STR, converts a number to a string. This version returns the output length, though.
	DATE MANIPULATION
CDTODD	Convert "MM/DD/YYYY" version of date into standard ddate *4 format.

Table 10-2. General-Purpose FORTRAN Utilities By Type

NAME	PURPOSE
CKDATE	Ensures a string contains a valid date representation. Can be called before to cdtodd to avoid errors. Logical function.
CKDATI	Checks a 3-integer version of a date to ensure it's valid (date in mm,dd,yy integers). Logical function.
CVTDAT	Converts a "MM/DD/YYYY" date into a 3-integer form, returning an error flag if it can't do it. This routine is obsolete; a combination of CKDATE, CDTODD, and DDTOID should be used instead.
DATEMK	Low-level date subsystem utility which converts the subsystem date representation (*4 Julian since 1601) into a 3-integer form. Meant to be called only by higher-level date utilities.
DATEP1	Increments a 3-integer version of a date by one day.
DATSTR	Returns today's date in a "MM/DD/YY" format.
DCLRFY	Takes a RELATE representation of a date (*4 word) and sets all unused bits to 0. Good insurance against date subsystem aborts, since RELATE appears to set these bits randomly, causing some of our routines to have problems.
DDATE	Integer*4 function returning today's date in the RELATE *4 format.
DDTOCD	Character*10 function which converts from a ddate format to "MM/DD/YYYY".
DDTOID	Converts from a ddate format to a 3-integer format.
DEARLY	Function returning true if first argument earlier than second (both arguments in ddate form).
ERLDAT	Returns the earliest possible ddate. This is an entry point in LATDAT.
FDDATE	Returns the first date in a given period in a ddate format, for a wide variety of period types.
GDATEP	Returns to first day of the i-th period in a ddate format, for a given fddate and period type.
GPERN	Given a ddate, returns the number of the period it falls in, for a given fddate and period type.

Table 10-2. General-Purpose FORTRAN Utilities By Type

NAME	PURPOSE
IDAYS	Integer function returning the number of days between two dates, specified in 3-integer format.
IDTODD	Converts from 3-integer format to ddate format.
JDAYS	Like IDAYS but returns a *4 number instead of *2.
LATDAT	Returns the latest possible ddate.
LMONTH	Integer function returning the number of days in a given month.
MRKDAY	Converts a 3-integer date into a ddate. Low-level utility meant to be called only by high-level date routines.
NWDATE	Integer*4 function returning a ddate N days later than a given ddate.
NWDATU	Integer*4 function returning the ddate N periods after a given ddate.
NWIDAT	Same as NWDATE but input and output in 3-integer format.
RDATE	Returns today's date in RELATE Real storage format. Obsolete, use DDATE instead.
RDFSTR	Converts from an RDATE format into a "MM/DD/YY" format. Obsolete, standardization mandates ddate formats.
	DEVICE CONTROL
LPSEND	Closes a spooled output file (opened with LPSET), causing actual printing to commence.
LPSET	Returns a FORTRAN i/o unit number opened on the device specified by the user in his user environment parameter menu.
SCLEAR	Clears the screen. Depends on the current terminal type setting on the user environment parameter menu being correct.
SETCCL	Reads the user environment parameter menu terminal type setting a stores the proper screen clear character sequence for use by SCLEAR.
SETTTY	Attempts to discover the user's terminal type by figuring out what port he's logged on through. Port number logic is hard-wired into the routine.

Table 10-2. General-Purpose FORTRAN Utilities By Type

NAME	PURPOSE
	DIAGNOSTICS
DEBUG	Logical function which reads the LPRNTON job control word. Generally used in a statement like IF (debug) CALL setlpr.
SETLPR	Prompts user for changes to current lprnts settings.
SLPRNT	Takes an lprnts array index and a true/false argument and sets that lprnt to that value.
	ERROR MESSAGE OUTPUT
ERRMSG	Writes a delimited text string to the screen, preceded by "*** ".
LABORT	Constructs an abort message which includes "AT line number" plus a user message. Useful when an input file is being processed and you want to tell the user what line the problem occured at.
LWARN	Like LABORT but just prints the message without aborting.
MABORT	Prints an error message contained in a delimited text string and calls ZABORT.
ZABORT	General abort routine. Prints an abort notice and STOPs execution.
	FILE OPEN/INPUT/OUTPUT
DWRITE DWRIT1 DREAD DREAD1	These are all entry points in DWRITE. They do direct access reads/writes of a specified record to a specified location on a specified unit number. The regular entries abort on an error, the "1" entries set an error flag and return.
FEXIST	Logical function which returns true if the file named in the argument exists.
FILCLS	Closes a file opened via FOPEN.
FILOPN	Opens a file for FORTRAN access. Files include devices in this context. See the text on file i/o for an exposition of all the possible file specifiers---any kind of file can be created/opened.

Table 10-2. General-Purpose FORTRAN Utilities By Type

NAME	PURPOSE
UREAD UWRITE	UWRITE is an entry point in UREAD. These do unformatted sequential-access reads/writes between a given integer array and an active unit number. No error checking.
	FORMATTERS
EJECT	Does a page eject on the given unit number.
PGINIT PGRSET PGSEND PGWRIT	These routines comprise the page printing system. They are in UTLO. You can set up a header and a page size and other attributes and just send lines to this system, letting it worry when to do the page breaks. See the text on formatters for a fuller description.
PRTHLP	This is useful for printing bulk help text or static menus. It expects a unit number which is connected to a sequential ASCII file in 80-column editor format, and a section header label. It reads through the file, finds the section by getting a match on the header, and prints the section. Much easier than putting things into format statements.
TRECOL	Prints a list (array) of character elements in three columns onto a specified unit.
	INITIALIZATION
CFINIT	Initializes the Core command system's stored commands subsystem. The routine MUST be called before the READLN utility can be used.
GETGRP	Determines whether the user is running the development or production version of ALIAS, and sets the variable that holds the group name where menu system files and relations will be.
INIPRC	Does general initialization for a FORTRAN module being executed as a son process by the Core, including swap-in of the Core common blocks generally of interest (e.g., /uzrprv/, /scenar/, and /pvalue/.
INIIOC INITIO	Together these routines will initialize i/o for the utilities and for a FORTRAN module in general. Mainly they set the integer variables which hold the standard input and output unit numbers. Mabort, zabort, etc. will not work if these are not called.

Table 10-2. General-Purpose FORTRAN Utilities By Type

NAME	PURPOSE
TTYINI	A terminal-type detection utility which works by querying the user. Sometimes useful during module debugging when you don't want to be hooked up to the Core; the screen clear sequence is placed in the /tty/ block.
	LINE INPUT
RDLN	Reads a 72-character line from a given unit without uppercasing.
RDLNC	Reads 72-character lines without uppercasing and keeps track of the number read in the /readc/ block.
RDLNCU	Like RDLNC but uppercases as well.
READLN	The main System Core line-read routine. This routine knows about the stored commands subsystem; it will automatically close a command file and reset to normal terminal i/o operation (via a call to stopcf) when the end of the command file is reached. Always use this routine for obtaining user input in the Core, and for any module linked into the Core that you want to be serviced by the stored commands subsystem. Note that READLN uppercases all input.
	MATHEMATICS
IXSUM	Sums up all elements of a 1-dimensional vector (array). Integer*2 function.
RANF RANGET RANSET RANST1 RANTRP	Random number generator. Provides numbers along up to 10 sequences; specify sequence when calling RANF. Initialization of seed for a specific sequence done by call to RANSET. RANST1 initializes all sequences. RANGET returns the current status of all sequences for saving.
VSUMNI	Vector sum for two 1-dimensional vectors.
VSUBNI	Vector difference for two 1-dimensional vectors. This is an entry point in VSUMNI.
	MEMORY MANAGER
FINMEM GETMEM	The memory manager supports use of extra data segments for extended global storage. Inimem initializes for a

Table 10-2. General-Purpose FORTRAN Utilities By Type

NAME	PURPOSE
INIMEM PUTMEM	given segment, putmem and getmem allow transfer of arrays between the segment and directly addressable memory, and finmem releases the segment. These utilities are very useful if your program requires more than 64K bytes of data memory. You can page the big arrays to one or more 64K segments.
	MISCELLA[illegible]S
LISTON	Given a menu name and a scenario key field value, this routine returns a list of candidates on the given list menu and their on/off statuses.
MODCOR	An alternative modulo function.
STOPCF	Really part of the Core command system's stored commands subsystem, this must be in the RL because it is called by READLN. The routine just resets i/o and command system units and flags on end-of-command-file.
	OPERATING SYSTEM INTERFACE
CPUTIM	Current system cpu clock time in milliseconds. Two calls (this is a real function) give an interval.
MONCOM	Executes a monitor command (i.e. an MPE command).
USRINF	Returns id information about the user, including name and log-on group. Has 3 entry points: USRNAM, USRGRP, USRACT.
	SORTING SEARCHING
CHASH	Does a hash-type sort on a character array. Does not actually sort the array, just returns an array of integers that give the sorted order of the character array elements.
CHASHV	Used by CHASH.
JHASH	Same as CHASH, but operates on a *4 integer array.
MATCH2	Integer function returning the location in an array of integers of a given target integer.
MATCHC	Same as MATCH2, looks for a match for a character string.

Table 10-2. General-Purpose FORTRAN Utilities By Type

NAME	PURPOSE
MTCHOC	Same as MATCHC but assumes that character array it is to search is sorted. More efficient if this is true.
QSORTC	Returns a character array sorted. Uses heap sort method.
	STACK DATA TYPE
CSINIT	See text discussion of the stack data type for purpose and organization of these routines. CSINIT initializes the stack system.
CSPOP	Pops an item off the stack
CSPOPR	Pops and returns an item from the stack.
CSPSH	Pushes an item onto the stack.
CSPSH2	Pushes two items onto the stack.
CSPSHL	Pushes many items onto the stack.
CSRD	Reads the top item on the stack and leaves it there.
CSRD2	Reads two items at top of stack and leaves them there.
CSRDL	Reads many items and leaves them there.
	USER INTERACTION
CNTINU	Stops execution and prompts the user to hit return to continue. Useful when a message has been put up and the user needs to have the leisure to read it before more output is done.
DPAUSE	Does an N-second process pause.
QUERY	Logical function prompting user for yes-or-no answer to a given (delimited) text query. Uses YESNO.
YESNO	Logical function which forces the user to answer Y, YES, N, or NO. Calls READLN for input, so cfinit and initio and iniioc must be called before this is used. Does not print prompt.

Table 10-3. General-Purpose FORTRAN Utilities By Name

NAME	SOURCE FILE	TYPE
ASCINT	UTLRA	DATA RANGE CHECKING
ASCPRN	UTLRA	DATA RANGE CHECKING
ASCREL	UTLRA	DATA RANGE CHECKING
BETWN	UTLRA	DATA RANGE CHECKING
CCAT2	UTLRA	CHARACTER STRING MANIPULATION
CCAT3	UTLRA	CHARACTER STRING MANIPULATION
CCAT4(e)	UTLRA	CHARACTER STRING MANIPULATION
CDTODD	UTLRA	DATE MANIPULATION
CEQ	UTLRA	CHARACTER STRING MANIPULATION
CFINIT	UTLRA	INITIALIZATION
CHASH	UTLRA	SORTING SEARCHING
CHASHV	UTLRA	SORTING SEARCHING
CHNALO	UTLRCHN	CHARACTER STRING MANIPULATION
CHNDEA	UTLRCHN	CHARACTER STRING MANIPULATION
CHNFRE	UTLRCHN	CHARACTER STRING MANIPULATION
CHNINI	UTLRCHN	CHARACTER STRING MANIPULATION
CIF	UTLRCI	CHARACTER STRING MANIPULATION
CKDATE	UTLRCI	DATE MANIPULATION
CKDATI	UTLRCI	DATE MANIPULATION
CNTINU	UTLRCI	USER INTERACTION
CPUTIM	UTLRCI	OPERATING SYSTEM INTERFACE
CRNGI	UTLRCI	DATA RANGE CHECKING
CRNGI4	UTLRCI	DATA RANGE CHECKING
CSINIT	UTLRCS	STACK DATA TYPE
CSPOP	UTLRCS	STACK DATA TYPE
CSPOPR	UTLRCS	STACK DATA TYPE
CSPSH	UTLRCS	STACK DATA TYPE
CSPSH2	UTLRCS	STACK DATA TYPE
CSPSHL	UTLRCS	STACK DATA TYPE
CSRD	UTLRCS	STACK DATA TYPE
CSRD2	UTLRCS	STACK DATA TYPE
CSRDL	UTLRCS	STACK DATA TYPE
CVTDAT	UTLRCV	DATE MANIPULATION
DATEMK	UTLRCV	DATE MANIPULATION
DATEP1	UTLRCV	DATE MANIPULATION
DATSTR	UTLRCV	DATE MANIPULATION
DCLRFY	UTLRCV	DATE MANIPULATION
DDATE	UTLRCV	DATE MANIPULATION
DDTOCD	UTLRCV	DATE MANIPULATION
DDTOID	UTLRCV	DATE MANIPULATION
DEARLY	UTLRCV	DATE MANIPULATION
DEBUG	UTLRCV	DIAGNOSTICS
DELIM	UTLRCV	CHARACTER STRING MANIPULATION
DINDEX	UTLRDI	CHARACTER STRING MANIPULATION
DPAUSE	UTLRDI	USER INTERACTION
DREAD(e)	UTLRDI	FILE OPEN/INPUT/OUTPUT
DREAD1(e)	UTLRDI	FILE OPEN/INPUT/OUTPUT
DWRIT1(e)	UTLRDI	FILE OPEN/INPUT/OUTPUT
DWRITE	UTLRDI	FILE OPEN/INPUT/OUTPUT
EJECT	UTLRDI	FORMATTERS

Table 10-3. General-Purpose FORTRAN Utilities By Name

NAME	SOURCE FILE	TYPE
ELIMBL	UTLRDI	CHARACTER STRING MANIPULATION
ERLDAT	UTLRDI	DATE MANIPULATION
ERRMSG	UTLRDI	ERROR MESSAGE OUTPUT
FDDATE	UTLRF	DATE MANIPULATION
FEXIST	UTLRF	FILE OPEN/INPUT/OUTPUT
FILCLS	UTLRF	FILE OPEN/INPUT/OUTPUT
FILOPN	UTLRF	FILE OPEN/INPUT/OUTPUT
FINMEM	UTLRFIN	MEMORY MANAGER
FLT	UTLRFIN	DATA TYPE CONVERSION
GDATEP	UTLRFIN	DATE MANIPULATION
GETGRP	UTLO	INITIALIZATION
GETMEM	UTLRFIN	MEMORY MANAGER
GPERN	UTLRFIN	DATE MANIPULATION
IAND	UTLRFIN	BIT MANIPULATION
IDAYS	UTLRFIN	DATE MANIPULATION
IDTODD	UTLRFIN	DATE MANIPULATION
INIIOC	UTLRFIN	INITIALIZATION
INIMEM	UTLRFIN	MEMORY MANAGER
INIPRC	UTLO	INITIALIZATION
INITIO	UTLRFIN	INITIALIZATION
IOR	UTLRFIN	BIT MANIPULATION
IXSUM	UTLRFIN	MATHEMATICS
JDAYS	UTLRFIN	DATE MANIPULATION
JHASH	UTLRFIN	SORTING SEARCHING
KFIX	UTLRK	DATA TYPE CONVERSION
LABORT	UTLRK	ERROR MESSAGE OUTPUT
LATDAT	UTLRK	DATE MANIPULATION
LBIT	UTLRK	BIT MANIPULATION
LETNUM	UTLRK	DATA RANGE CHECKING
LETONL	UTLRK	DATA RANGE CHECKING
LISTON	UTLO	MISCELLANEOUS
LMONTH	UTLRK	DATE MANIPULATION
LOWERC(e)	UTLRK	CHARACTER STRING MANIPULATION
LPSEND	RECOMP	DEVICE CONTROL
LPSET	RECOMP	DEVICE CONTROL
LSTRNG	UTLRK	CHARACTER STRING MANIPULATION
LTRIM	UTLRK	CHARACTER STRING MANIPULATION
LWARN	UTLRK	ERROR MESSAGE OUTPUT
MABORT	UTLRM	ERROR MESSAGE OUTPUT
MATCH2	UTLRM	SORTING SEARCHING
MATCHC	UTLRM	SORTING SEARCHING
MODCOR	UTLRM	MISCELLANEOUS
MONCOM	UTLRM	OPERATING SYSTEM INTERFACE
MPDCOD	UTLRM	CHARACTER STRING MANIPULATION
MRKDAY	UTLRM	DATE MANIPULATION
MSTRNG(e)	UTLRM	CHARACTER STRING MANIPULATION
MTCHOC	UTLRM	SORTING SEARCHING
NCFLW(e)	UTLRM	CHARACTER STRING MANIPULATION
NCFW	UTLRM	CHARACTER STRING MANIPULATION
NCSW(e)	UTLRM	CHARACTER STRING MANIPULATION

Table 10-3. General-Purpose FORTRAN Utilities By Name

NAME	SOURCE FILE	TYPE
NLWFC(e)	UTLRM	CHARACTER STRING MANIPULATION
NSWFC(e)	UTLRM	CHARACTER STRING MANIPULATION
NSWFW(e)	UTLRM	CHARACTER STRING MANIPULATION
NUMASK	UTLRM	DATA TYPE CONVERSION
NUMSFX	UTLRM	DATA TYPE CONVERSION
NWDATE	UTLRNW	DATE MANIPULATION
NWDATU	UTLRNW	DATE MANIPULATION
NWFC(e)	UTLRNW	CHARACTER STRING MANIPULATION
NWFSW(e)	UTLRNW	CHARACTER STRING MANIPULATION
NWIDAT	UTLRNW	DATE MANIPULATION
PGINIT	UTLO	FORMATTERS
PGRSET	UTLO	FORMATTERS
PGSEND	UTLO	FORMATTERS
PGWRIT	UTLO	FORMATTERS
PLURAL	UTLRNW	DATA TYPE CONVERSION
PRTHLP	UTLRNW	FORMATTERS
PUTMEM	UTLRNW	MEMORY MANAGER
QSORTC	UTLRQ	SORTING SEARCHING
QUERY	UTLRQ	USER INTERACTION
RANF	UTLRQ	MATHEMATICS
RANGET(e)	UTLRQ	MATHEMATICS
RANSET(e)	UTLRQ	MATHEMATICS
RANST1(e)	UTLRQ	MATHEMATICS
RANTRP	UTLRQ	MATHEMATICS
RDATE	UTLRQ	DATE MANIPULATION
RDFSTR	UTLRQ	DATE MANIPULATION
RDLN	UTLO	LINE INPUT
RDLNC	UTLO	LINE INPUT
RDLNCU	UTLO	LINE INPUT
READLN	UTLRQ	LINE INPUT
RTRIM	UTLRQ	CHARACTER STRING MANIPULATION
SCLEAR	RECOMP	DEVICE CONTROL
SETCCL	RECOMP	DEVICE CONTROL
SETLPR	UTLRS	DIAGNOSTICS
SETTTY	RECOMP	DEVICE CONTROL
SLPRNT	UTLRS	DIAGNOSTICS
STOPCF	UTLRS	MISCELLANEOUS
STRN	UTLRS	DATA TYPE CONVERSION
TRECOL	UTLRS	FORMATTERS
TTYINI	UTLRS	INITIALIZATION
UPPERC	UTLRS	CHARACTER STRING MANIPULATION
UREAD	UTLRS	FILE OPEN/INPUT/OUTPUT
USRACT	UTLRS	OPERATING SYSTEM INTERFACE
USRGRP	UTLRS	OPERATING SYSTEM INTERFACE
USRINF	UTLRS	OPERATING SYSTEM INTERFACE
USRNAM	UTLRS	OPERATING SYSTEM INTERFACE
UWRITE(e)	UTLRS	FILE OPEN/INPUT/OUTPUT
VSUBNI(e)	UTLRS	MATHEMATICS
VSUMNI	UTLRS	MATHEMATICS
XMIT/B	UTLRX	DATA MOVEMENT

Table 10-3. General-Purpose FORTRAN Utilities By Name

NAME	SOURCE FILE	TYPE
XMIT2/B	UTLRX	DATA MOVEMENT
XMIT4/B	UTLRX	DATA MOVEMENT
XMITC	UTLRX	DATA MOVEMENT
YESNO	UTLRX	USER INTERACTION
ZABORT	UTLRX	ERROR MESSAGE OUTPUT

10.1.1 Discussion By Type of Utility

This section will discuss some (not all) of the utility types listed in Table 10-1. Most are self-explanatory, but some exist because of particular features of the HP or of RELATE; these features need to be elucidated.

10.1.1.1 Character String Utilities

Many of the existing character string oriented utilities would be unnecessary if the HP 3000 had an ANSI 1977 standard FORTRAN compiler. Concatenation (CCAT_ routines) would be done using FORTRAN syntax, and the string chain data type (CHN___ routines) would be less necessary because strings would not be limited to 255 character lengths. The various delimit-undelimit utilities wouldn't be needed.

10.1.1.2 Date Manipulation Utilities

Most ALIAS modules must work with dates. The extensive array of ALIAS date utilities makes this a straightforward rather than maddening task by allowing the programmer to convert between many date formats, to compare dates and calculate intervals, and to work in terms of periods (months, weeks, years, etc.) as well as days.

A date can be stored and/or manipulated in four formats:

1) 10-character ASCII (MM/DD/YYYY), convenient for user i/o.

2) 3-integer format, i.e. 3 two-byte integers each holding one of month, day, year.

3) RELATE double-integer word (D) date storage format. RELATE stores a date in a double integer by reserving ranges of bits within the 32-bit word for the month, day, and year quantities.

4) "Ddate" format. This is the date utilities' preferred format. Stored in a double integer word, dates are expressed in Julian form with a basis date of 31 Dec 1600.

Although the large variety of formats may seem unwieldly, the first two are very convenient, and the last two are necessary.

10.1.1.3 Device Control

ALIAS programs typically want to work with two kinds of device: the user's terminal, and spooled printers. The only action supported with respect to terminals is a screen-clear. Programs which require sophisticated screen management should be written in BUILDER if possible.

The screen-clear utilities consult the contents of the TTYTYP parameter on the User Environment Parameters menu of the Command System every time a clear is requested. They contain hard-wired logic which converts the code names found there into screen-clear character sequences. Should additional terminal types come into use on ALIAS, the code of these routines will need to be changed.

10.1.1.4 File Open/Input/Output

One of the major weaknesses of HP FORTRAN is its lack of an OPEN statement. Dynamic opening of a file involves some very messy calls to MPE Intrinsic routines. To make direct use of these avoidable, the filopn/filcls utilities were written. Filopn takes three arguments: the unit number the file should be opened on, a logical flag which is returned .true. if the operation succeeded, and a delimited string of directives separated by commas. Syntax of a typical filopn call might be:

```
CALL filopn(unit,ok,"+name=myfil.grp,new,ascii,write,fixlrecs+")
```

which requests creation of a new ascii fixed record length file in the .grp group named myfil.

Table 10-4 gives all directives that filopn accepts. Filcls requires similar arguments, but has directive options of

Table 10-4. FILOPN Directives

DIRECTIVE	DEFAULT VALUE	EFFECT

NAME=filnam
DEVICE=##
	FTN##	Specifies name of file to open. If name does not include a group, log-on group is assumed.

NEW OLD OLDTEMP	NEW	Use NEW to create a new file, OLD to open an existing permanent file, OLDTEMP for an existing temporary file.
READ WRITE APPEND UPDATE READWRITE	READ	Specifies the types of operations you will be allowed to perform on the file.
EXCLUSIVE SHARED LOCKABLE	SHARED if READ else EXCLUSIVE	LOCKABLE applies only if SHARED SHARED not recommended when you are going to write to the file. EXCLUSIVE will cause an error message if someone else already has it open, but no abort. Detect open failure by flag returned by filopn.
ASCII BINARY	BINARY	You must choose the form of the file when it is created. Binary files may not be edited. Ignored if file is OLD.
KSAM SEQUENTIAL	SEQUENTIAL	No one knows how to use KSAM.
FIXLRECS VARLRECS	VARLRECS	Variable-length records save space, especially when the file is ASCII, but the editors work best with fixed-length records.

RECLEN=####
MAXRECS=#####
RECPBLK=###
	133
	1023
	system decides
		Record length in bytes.
		Maximum number records in file.
		Blocking factor.

LABELED		Indicates a labeled tape. Filopn not so far used with tape.

SAVE, DELETE, and TEMPORARY only. Note that every filopn call directive list except those including DEVICE= must specify a file name and an access type (e.g. READ or WRITE).

Note that a file opened as "NEW" will not in fact be created in the permanent file domain until successfully closed by a filcls call with the "SAVE" option.

The utilities which perform direct-access file i/o are convenient because they automatically perform error checking during the read/write, producing either a "nice" abort or else returning a status flag to the caller.

10.1.1.5 Formatters

The output formatting utilities can be extremely useful. The prthlp routine is used by the scenario system for display of menus, and by the assigner for display of all help text. It is a good means for shoving large volumes of text at the user.

The PG ---- routines form a subsystem that can make production of reports much easier from FORTRAN. A common problem in report generation is the necessity to count output lines so that page ejects can be given at appropriate points, and so that headers can be written at the top of each page. Also, it is often desirable to prompt the user before each page eject when output is going to the screen. The PG routines handle the details of all of this for the programmer. Output can be generated and sent to the subsystem line by line with no worries.

To use the subsystem, call pginit and pgrset. Pgrset can be called at any time to begin output of a new report. Pgrset wants such things as the unit number of the output file or device, its record length and page length, the formfeed character, and the mode the PG subsystem should operate in.

The subsystem works by storing each line sent to it in a buffer until it has a full page. Lines are sent by calls to the PGSEND routine, whose arguments include a page header of as many lines as the developer chooses. Its action when the page is full depends on the operating mode. There are four mode choices, specified by number:

1) PG routines send output to unit when buffer has a full page, user is prompted before output sent. This is most appropriate for screen output.

2) Same as 1 but output is continuous, user is not prompted. Most appropriate for line printer output.

3) Same as 2 but the header (specified as argument to PGSEND) is printed only at the top of the first page, not at the top of every page as in 1 and 2.

4) PGSEND does not send output to the unit automatically. Instead, it returns a flag when there is a full page in the buffer, leaving it to the user to call PGWRIT to print the buffer contents.

The variety of operating modes makes the subsystem configurable to most situations.

10.1.1.6 Initialization

Programs which intend to use any of the utilities should always call, in this order, the INITIO, INIIOC, and CFINIT initialization routines. These set certain key global i/o variables, mostly unit numbers, which are relied on by some utilities.

The convenience of using iniprc to initialize FORTRAN modules executed as son processes was discussed in Sections 8 and 9.

10.1.1.7 Line Input

The READLN routine MUST be used to retrieve ALL terminal input in System Core routines which are to be serviced by the stored commands subsystem. This ensures that i/o redirection takes place properly. The other line-read routines can be useful when processing a text input file, since some of them will keep a running count of the number of lines read for use in error or progress messages.

10.1.1.8 Stack Data Type

The stack data type was discussed in Section 3.1.6 and in Section 8. The stack utility routines are currently capable of implementing only one stack per process; the stack is reserved for use by the Command System in the System Core process. However, extension of the utilities to manage multiple stacks would be straightforward.

10.1.2 General-Purpose FORTRAN Utility Abstracts

```
C     ASCINT ************************************************
$control segment=dmaint
      LOGICAL FUNCTION ascint(string,len)
      integer len
      character*(len) string
C*                                               *** ABSTRACT ***
C#PURPOSE Checks a string to be sure it contains only numbers.
C#AUDIT HISTORY
C        MSCarey          17-mar-83  AUTHOR
C#FORMAL PARAMETERS
Cin      len      length of input string
Cin      string   sting to be checked
C#COMMON BLOCKS
C        none
C#CALLER various
C#METHOD
C  Makes sure each byte is within proper octal range. Allows a
C  trailing blank for strings of odd length, ad a leading "-".
C#LOCAL VARIABLES
C        buffer   word-aligned version of input string
C##
```

```
C     ASCPRN ******************************************
$control segment=dmaint
      LOGICAL FUNCTION ascprn(string,len)
      integer len
      character*(len) string
C*                                           *** ABSTRACT ***
C#PURPOSE Checks a string to be sure it contains only printing chars
C#AUDIT HISTORY
C        MSCarey          17-mar-83  AUTHOR
C#FORMAL PARAMETERS
Cin      len      length of input tring
Cin      string   string to be checked
C#COMMON BLOCKS
C        none
C#CALLER various
C#METHOD
C  Makes sure that each byte is within the proper octal range.
C#LOCAL VARIABLES
C        buffer   word-aligned version of string
C##
```

```
C     ASCREL **********************************************
$control segment=dmaint
      LOGICAL FUNCTION ascrel(string,len)
      integer len
      character*(len) string
C*                                          *** ABSTRACT ***
C#PURPOSE Checks a string to be sure it contains only numbers or .
C#AUDIT HISTORY
C        MSCarey          17-mar-83  AUTHOR
C#FORMAL PARAMETERS
Cin      len      length of input string
Cin      string   string to be checked
C#COMMON BLOCKS
C        none
C#CALLER various
C#METHOD
C  Makes sure each byte is within allowed octal range, or is a "."
C  or a "-".
C#LOCAL VARIABLES
C        buffer   word-aligned version of string
C##
```

```
C     BETWN*****************************************************
      LOGICAL FUNCTION betwn(i,low,high)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer i,low,high
C*                                                 *** ABSTRACT ***
C#PURPOSE betwn  :=  low <= i <= high
C#AUDIT HISTORY
C         Densmore        04-Feb-83  AUTHOR
C#TYPE    misc. utility
C#FORMAL PARAMETERS
Cin       i,low,high   *2 integers for function
C##
```

```
C     CCAT2*****************************************************
$CONTROL check=2
      SUBROUTINE ccat2(s1,len1,s2,len2,sr,lenr,mlenr)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER len1,len2,lenr,mlenr
      CHARACTER*1 s1(len1), s2(len2), sr(mlenr)
C*                                          *** ABSTRACT ***
C#PURPOSE concatenates s1 to s2 and returns result in sr
C#AUDIT HISTORY
C         Densmore         15-Dec-82  AUTHOR
C#TYPE    string manipulation utility
C#FORMAL PARAMETERS
Cin       s1      first string, may be DTS
Cin       len1    length in characters of s1
Cin       s2      second string, may be DTS
Cin       len2    length of s2
Cout      sr      returned string; may have same address as s1 or s2
Cout      lenr    length of sr
Cin       mlenr   maximum length allowable for sr
C#METHOD
C  DTS refers to a Delimited Text String, in which the length
C  is determined by delimiters, one before and one following the
C  intended string.  The delimiter character is the first in the string
C  which is nonblank.:    '=abcdef=' --> 'abcdef'.  A string is assumed
C  to be DTS if and only if the length associated with it is ZERO.
C#LOCAL VARIABLES
C         b?      beginning position of (possibly delimited) string
C         e?      ending position of (possibly delimited) string
C         l?      length of (possibly delimited) string
C##
```

```
C     CCAT3*******ccat4*****************************************
$CONTROL check=2
      SUBROUTINE ccat3(s1,len1,s2,len2,s3,len3,sr,lenr,mlenr)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER len1,len2,len3,len4,lenr,mlenr
      CHARACTER*255 s1,s2,s3,s4,sr
C*                                            *** ABSTRACT ***
C#PURPOSE performs  sr := s1 !! s2 !! s3
C#AUDIT HISTORY
C         Densmore         15-Dec-82  AUTHOR
C#TYPE    string manipulation utility
C#FORMAL PARAMETERS
C:in      sN      strings
C:in      lenN    character lengths
Cout      sr      returned string
Cout      lenr    its length
Cin       mlenr   maximum length allowable for sr
C##
```

```
C     CDTODD**************************************************
$CONTROL check=3
      INTEGER*4 FUNCTION cdtodd(datstr)
      character*10 datstr
C*                                              *** ABSTRACT ***
C#PURPOSE Character Date TO Relate Date.  Converts a character
C  string of the form MM/DD/YYYY to a I*4  stored as
C  in RELATE format.  See the data type text in TDDATE.INCL
C#AUDIT HISTORY
C        MSCarey           16-mar-83  AUTHOR
C        Densmore          26-Apr-83  Moved from DMUTIL to UTILA
C#FORMAL PARAMETERS
Cin      datstr   date string
C#COMMON BLOCKS
C        none
C#CALLER various
C#METHOD
C  Parses the string, converts its parts, and places it in storage.
C##
```

```
C     CEQ*****************************************************
$CONTROL check=2
      LOGICAL FUNCTION ceq(str1,len1,str2,len2)
      INTEGER len1,len2
      CHARACTER str1*(len1), str2*(len2)
C*                                             *** ABSTRACT ***
C#PURPOSE compares two strings, padding on the right with blanks
C         since HP's Fortran does string comparisons differently
C#AUDIT HISTORY
C        MSCarey           2-Feb-83  AUTHOR
C#TYPE   character utility
C#FORMAL PARAMETERS
Cin      str1     left character string in comparison
Cin      len1     length of str1
Cin      str2     right character string
Cin      len2     length of str2
C#COMMON BLOCKS
C        none
C#METHOD
C   one line routine. necessary because HP will not consider
C   two identical strings of different length to be equal.
C   Assumes that both strings are left-justified.
C##
```

```
C     CFINIT**************************************************
$CONTROL SEGMENT=MENU
      SUBROUTINE cfinit
C*                    *** FORMAL PARAMETER DECLARATIONS ***
C*                                         *** ABSTRACT ***
C#PURPOSE Command Files INITialize.  Initializes comcfl common
C         block switches to false.
C#AUDIT HISTORY
C         MSCarey         14-FEB-83  AUTHOR
C#FORMAL PARAMETERS
C         NONE
C#COMMON BLOCKS
Cout      comcfl  command file facility switches and io assignments
C#CALLER  mnurun
C#METHOD
C  Assignment statements.
C#LOCAL VARIABLES
C        none
C##
```

```
C     CHASH*******************************************************
$CONTROL check=3
      SUBROUTINE chash(a,len,kmax,nrec,k,amin,amax,ih,nh)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER len,kmax,nrec,k,nh, ih(nh)
      CHARACTER*(len) a(kmax,nrec), amin, amax
C*                                            *** ABSTRACT ***
C#PURPOSE Returns the sorted order of the char records A based on row k
C#AUDIT HISTORY
C          Densmore       19-Jun-83  AUTHOR
C#TYPE    Sort utility
C#FORMAL PARAMETERS
Cin       a       the array of nrec records, each of length kmax
Cin       len     the number of chars in each element of each rcd
Cin       kmax    the number of elements in each record
Cin       nrec    the number of records
Cin       k       the element of each record on which to sort
Cin       amin    a lower bound on the values a(k,*)
Cin       amax    an upper bound on the values a(k,*)
Cout      ih      the sorted order of the records contained in a,
C                 based on the element k in each record.  That is,
C                 ih(1) contains the number of the record which
C                 appears first when they are given in order; ih(2)
C                 contains the number of the second record, etc.
C         nh      the length of the array ih.  This number must
C                 obviously be >= nrec; ih is actually used as a
C                 work area and should be at least 2*nrec, preferably
C                 3*nrec.
C#METHOD
C  Assumes that the records are approximately linearly distributed.
C  Takes the value of each record's key and uses it to estimate its
C  sequence number, placing that record's index number in the ih
C  element corresponding to that sequence number.  This is repeated
C  for each record, resolving collisions as required.  If only a
C  few collisions need to be resolved this is a nearly linear order-
C  ing algorithm.  At the end the nonzero (unfilled) elements of the
C  ih array are removed and the filled elements left shifted so that
C  the first nrec elements of ih give the ordering information.
C##
```

```
C     CHASHV*************************************************
$CONTROL check=2
      REAL FUNCTION chashv(s,m)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER m
      CHARACTER*1 s(m)
C*                                              *** ABSTRACT ***
C#PURPOSE Returns hash value from short char string for chash
C#AUDIT HISTORY
C         Densmore        20-Jun-83  AUTHOR
C#TYPE    sort utility
C#FORMAL PARAMETERS
Cin       s       the character string
Cin       m       its length
C#CALLER  chash
C#METHOD
C  Pretends each character is a base-128 digit;  assumes blank
C  padding exists in s and that m has the same positive value for
C  every call from a given invocation of chash.
C##
```

```
C     CHNALO**************************************************
$CONTROL check=3
      INTEGER FUNCTION chnalo(chn)
      PARAMETER nhd=2
      INTEGER chn(nhd)
C#PURPOSE allocates from chain system.  See chnini.
C##


C     CHNDEA**************************************************
$CONTROL check=3
      SUBROUTINE chndea(chn,item)
      PARAMETER nhd=2
      INTEGER chn(nhd),item
C#PURPOSE deallocates item back into available area of chain
C         system.  See chnini.
C##


C     CHNFRE**************************************************
$CONTROL check=3
      LOGICAL FUNCTION chnfre(chn,item)
      PARAMETER nhd=2
      INTEGER chn(nhd),item
C#PURPOSE checks availability of item in chain.  See chnini.
C##
```

```
C     CHNINI***************************************************
$CONTROL check=3
      SUBROUTINE chnini(chn,size,nitems)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER size,nitems,chn(size)
C*                                          *** ABSTRACT ***
C#PURPOSE initializes chain allocation system
C#AUDIT HISTORY
C         Densmore        09-Dec-82  AUTHOR
C#TYPE    chain utility
C#FORMAL PARAMETERS
Cout      chn     chain array of length nitems+2; the first word
C                 contains a copy of nitems, the second contains
C                 the address of the first available item value.
C                 The next is initialized to the first item value.
Cin       size    the length of the chn array
Cin       nitems  the number of items to be chained
C#METHOD
C  For a call CALL chnini(chn,size,nitems), this routine initializes
C  the chn array so that it contains (nitems,1, 1,2,3,4,...,nitems)
C  Accompanying this routine are three others: chnalo to allocate
C  new item values, chndea to deallocate item values by name, and
C  chnfre to verify that an item value is free (not normally needed
C  by the "outside world". Once an item value is allocated, the
C  idea is that the item value is never again returned by chnalo
C  unless at some future time that item value is deallocated.  These
C  routines can be used in conjunction with a doubly dimensioned
C  array which the item values may then serve as indexes.
C
C  Chain initialization: CHNINI(chn,size,nitems)
C  Chain allocation:     CHNALO(chn) integer fcn returns item value
C  Chain deallocation:   CHNDEA(chn,item)
C  Chain free item test: CHNFRE(chn,item) logical fcn
C     returns true if item is available for allocation
C
C##
```

```
C     CIF*****************************************************
$CONTROL check=2
      LOGICAL FUNCTION cif(str1,len1,str2,len2)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER len1,len2
      CHARACTER str1*(len1), str2*(len2)
C*                                            *** ABSTRACT ***
C#PURPOSE performs complete blank strip on both strings; then compares
C#AUDIT HISTORY
C         Densmore         04-Feb-83  AUTHOR
C#TYPE    character utility
C#FORMAL PARAMETERS
Cin       str?    the strings to compare
Cin       len?    the lengths of each string
C#LOCAL VARIABLES
C  lft?  leftmost nonblank
C  lgth? length between leftmost and rightmost nonblank
C##
```

```
C     CKDATE*************************************************
$CONTROL check=2
      LOGICAL FUNCTION ckdate(string,length)
      integer length
      character*(length) string
C*                                            *** ABSTRACT ***
C#PURPOSE Checks that string is valid character representation of a date
C#AUDIT HISTORY
C        MSCarey          17-mar-83  AUTHOR
C        Densmore         28-Apr-83  Extensive mod to check date
C        Densmore         14-Oct-83  change lmonth data to Fn ckdati
C#FORMAL PARAMETERS
Cin      length   length of input string
Cin      string   string to be checked
C#CALLER various
C#METHOD
C  First checks that all characters in the string are blanks,
C  digits, or slashes; further that there are exactly two slashes.
C  Then the respective numbers are checked for validity.
C##
```

```
C     CKDATI**************************************************
      LOGICAL FUNCTION ckdati(month,day,year)
      INTEGER month,day,year
C#PURPOSE Finishes work of ckdate; separate entry point in
C         case the work needs to be done from the intermediate
C         step of (mm,dd,yy)
C#AUTHOR  Densmore   14-Oct-1983
C#FORMAL PARAMETERS
Cin       month,day,year - INTEGER input date to check
CFunction ckdati         - LOGICAL true if date is a valid one
C##
```

```
C     CNTINU*************************************************
$CONTROL segment=seg'
      SUBROUTINE  cntinu
C*                                          *** ABSTRACT ***
C#PURPOSE   Pauses execution until user hits return.
C     Main use is ensuring that error messages stay on screen.
C     USES READLN, requires cfinit call befor usage.
C#AUDIT HISTORY
C        MSCarey        14-dec-83  AUTHOR
C#FORMAL PARAMETERS
C        none
C#COMMON BLOCKS
Cin      ioc      system io units
C#CALLER various
C#METHOD
C     fortran write; call to readln for read
C#LOCAL VARIABLES
C        line     readln argument
C        eof      readln argument
C##
```

```
C     CPUTIM*************************************************
      REAL FUNCTION cputim(dummy)
C*                          *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER dummy
C*                                                *** ABSTRACT ***
C#PURPOSE Returns a CPU second value directly from the internal clock
C         (value MAY NOT BE NORMALIZED to any zero)
C#AUDIT HISTORY
C         Densmore        27-Oct-82  AUTHOR
C#TYPE    Simple function
C#FORMAL PARAMETERS
C#COMMON BLOCKS
Cin       dummy   not used...just allows the function call
Cfunction cputim  output real value...must be normalized
C                 to be of use by saving the value returned
C                 by the first call.
C#METHOD
C  Calls the PROCTIME intrinsic, which gives a doubleword in millisecs
C##
```

```
C     CRNGI****************************************************
      SUBROUTINE crngi(i,low,high,text)
      INTEGER i,low,high
      CHARACTER*255 text
C#PURPOSE checks that low<=i<=high for INTEGER*2 variables
C#AUDIT HISTORY
C         Densmore  28-Oct-82  AUTHOR
C#FORMAL PARAMETERS
Cin       i         value
Cin       low       lowest possible value
Cin       high      highest possible value
Cin       text      delimited text string giving caller, etc.
C##
```

```
C     CRNGI4*****************************************************
      SUBROUTINE crngi4(i,low,high,text)
      INTEGER*4 i,low,high
      CHARACTER*255 text
C#PURPOSE checks that low<=i<=high for INTEGER*2 variables
C#AUDIT HISTORY
C         Densmore  28-Oct-82  AUTHOR
C#FORMAL PARAMETERS
Cin       i         value
Cin       low       lowest possible value
Cin       high      highest possible value
Cin       text      delimited text string giving caller, etc.
C##
```

```
C     CSINIT*******************************************************
      SUBROUTINE csinit
C#PURPOSE  command stack initializer
C  the routines documented here handle a command stack used to implement
C  recursive commands.
C#AUTHOR   Kerchner   December 14, 1981
C#TYPE     Command Stack Utility
C#FORMAL PARAMETERS
Cin  n         i*2  number of item referred to on stack
Cin  ni        i*2  number of items referred to in an array of it
Cin  item      i*4  item to be pushed onto top of stack
Cio  array     i*4  list of items pushed onto or read from stack
C#ROUTINE DEFINITIONS
C  n: int*2 length     a: typeless*4 item     A: typeless*4 array
C
C  SUBROUTINE csinit          - initializes command stack system
C  SUBROUTINE cspop(n)        - pops n items off stack (w/ no read)
C  SUBROUTINE cspopr(n,A)     - pops n items off stack into array A
C  SUBROUTINE cspsh(a)        - pushes the item a onto the stack
C  SUBROUTINE cspsh2(a2)      - pushes int*2 item a2 onto stack
C  SUBROUTINE cspshl(n,A)     - pushes the n items in A onto stack
C  INTEGER*4 FUNCTION csrd(n)- returns the n'th item on the stack;
C                               n=1 yields the top of the stack
C  INTEGER*2 FUNCTION csrd2(n)-returns n'th item as a *2 integer
C  SUBROUTINE csrdl(n,A)      - returns to array A the top n items
C
C  cspsh, cspsh2, and cspshl will overflow if stack array lacks room.
C  cspopr, csrd, csrd2, and csrdl will underflow if too many items read.
C  Subroutine cspop never underflows.
C##
```

```
C      CSPOP*****************************************************
       SUBROUTINE cspop(n)
       INTEGER n
C#PURPOSE  pops n items off stack; see csinit for complete documentation
C#AUTHOR   Kerchner   December 14, 1981
C#TYPE     Command Stack Utility
C##
```

```
C      CSPOPR****************************************************
$CONTROL check=2
       SUBROUTINE cspopr(ni,array)
       INTEGER ni
       INTEGER*4 array(ni)
C#PURPOSE  pops ni items off stack into array. complete doc in csinit
C#AUTHOR   Kerchner   December 14, 1981
C#TYPE     Command Stack Utility
C##
```

```
C      CSPSH*****************************************************
$CONTROL check=2
       SUBROUTINE cspsh(item)
       INTEGER*4 item
C#PURPOSE  pushes item onto top of stack. complete doc in csinit
C#AUTHOR   Kerchner   December 14, 1981
C#TYPE     Command Stack Utility
C##
```

```
C      CSPSH2****************************************************
$CONTROL check=2
       SUBROUTINE cspsh2(item2)
       INTEGER*2 item2
C#PURPOSE  pushes item2 onto top of stack. complete doc in csinit
C#AUTHOR   Densmore    3 Feb 1983
C#TYPE     Command Stack Utility
C##
```

```
C     CSPSHL***********************************************************
$CONTROL check=2
      SUBROUTINE cspshl(ni,array)
      INTEGER ni

      INTEGER*4 array(ni)
C#PURPOSE  pushes ni items in array onto stack.  complete doc in csinit
C#AUTHOR   Kerchner   December 14, 1981
C#TYPE     Command Stack Utility
C##


C     CSRD*************************************************************
$CONTROL check=0
      INTEGER*4 FUNCTION csrd(n)
      INTEGER n
C#PURPOSE  returns n'th item on stack;
C         see csinit for complete documentation.
C#AUTHOR   Kerchner   December 14, 1981
C#TYPE     Command Stack Utility
C##


C     CSRD2************************************************************
      INTEGER*2 FUNCTION csrd2(n)
      INTEGER n
C#PURPOSE  returns n'th item on stack, assuming it to be a *2 int;
C         see csinit for complete documentation.
C#AUTHOR   Densmore   3 Feb 1983
C#TYPE     Command Stack Utility
C##


C     CSRDL************************************************************
$CONTROL check=2
      SUBROUTINE csrdl(ni,array)
      INTEGER ni
      INTEGER*4 array(ni)
C#PURPOSE  reads top ni items on stack, nondestructively.
C         Complete doc in csinit.
C#AUTHOR   Kerchner   December 14, 1981
C#TYPE     Command Stack Utility
C##
```

```
C     CVTDAT****************************************
$CONTROL SEGMENT=MENU
       SUBROUTINE CVTDAT (INSTR ,IM,ID,IY,ERR)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR

       CHARACTER INSTR *LLINE
       LOGICAL ERR
       INTEGER IM,ID,IY
C*                                          *** ABSTRACT ***
C#PURPOSE decodes string of form 'month/day/year',
C         where year is specified as 1982 for example,
C         into IM=month, ID=day, IY=year.
C
C#AUDIT HISTORY
C         MEMutchler      17 JAN 83  AUTHOR
C         MEMutchler       7 FEB 83 TESTER  (program tstdat)
C#TYPE    mnurun utility
C#FORMAL PARAMETERS
Cin       instr   date string of form mm/dd/yyyy
Cout      im      month from 1 to 12
Cout      id      day   from 1 to 32
Cout      iy      year  as in 1983
Cout      err     true iff string not of correct form
C#COMMON BLOCKS
Cin       incpar  global parameter statements
C#CALLER  dpmenu
C#METHOD  split 'string' into month, day, and year pieces
C         by keying on '/', the delimeter, and convert them
C         to their integer values.
C#LOCAL VARIABLES
C         string  local of instr
C         lenstr  length of 'string' in non-blank characters
C         delim   character representation of '/'
C         piece   character string holding piece to be
C                 converted to integer.
C ##
C*                    *** INCLUDES and LOCAL DECLARATIONS ***
       INTEGER LENSTR , I
       CHARACTER PIECE*4, DELIM*1, STRING*LLINE
       DATA DELIM/'/'/
C*ENDDEC                            *** END DECLARATIONS ***
```

```
C     DATEMK*************************************************
$CONTROL check=3
      SUBROUTINE datemk(imark,month,day,year)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER*4 imark
      INTEGER month,day,year
C*                                              *** ABSTRACT ***
C#PURPOSE Given a mark day (mrkday), returns date (mrkday's inverse)
C#AUDIT HISTORY
C         Densmore        31-May-83  AUTHOR
C#TYPE    date utility
C#FORMAL PARAMETERS
Cin       imark   input mark day -- number of days since 31-Dec-1600
Cout      month   month corresponding to mark day (1-12)
Cout      day     day corresponding to mark day (1-31)
Cout      year    year corresponding to mark day
C#CONSTANTS
C         mark for 31-Dec-1600 is defined to be zero
C         mark for 31-Dec-2399 is the max allowable: maxmrk=291828
C         mark for 31-Dec-1999 is mk1999=145731 -- year 2000 is the
C                              only century leap year
C         mark for 31-Dec-2000 is mk2000=146097  -- same comment
C         number of days from 31-Dec-NN00 to 31-Dec-(NN+1)00
C                              is nd100=36524
C         number of days in a 4-year period is nd4=1461
C##
```

```
C     DATEP1***********************************************************
$CONTROL check=3
      SUBROUTINE datep1(m,d,y)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER m,d,y
C*                                                 *** ABSTRACT ***
C#PURPOSE adds one day to the date
C#AUDIT HISTORY
C         Densmore        01-Jun-83  AUTHOR
C#TYPE    date utility
C#FORMAL PARAMETERS
Cin/out   m,d,y   month (1..12), day(1..31), year incremented
C##
```

```
C     DATSTR**************************************************
      SUBROUTINE datstr(string)
      character*8 string
C*                                            *** ABSTRACT ***
C#PURPOSE returns current date as MM/DD/YY
C#AUDIT HISTORY
C       MSCarey          02-feb-83  AUTHOR
C#FORMAL PARAMETERS
Cout     string  see purpose
C#COMMON BLOCKS
C        none
C#CALLER  various
C#METHOD
C  Calls system intrinsic dateline and decode char month to num
C#LOCAL VARIABLES
C       bytara   argument for intrinsic call
C       month    holds strings for comparison against bytara
C       mnum     number of the month returned by dateline
C##
```

```
C     DCLRFY*************************************************
$CONTROL check=3
      INTEGER*4 FUNCTION dclrfy(rawdat)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER*4 rawdat
C*                                            *** ABSTRACT ***
C#PURPOSE Clarifies a raw RELATE DDATE.  See explanation
C         given in the data type include file TDDATE
C#AUDIT HISTORY
C         Densmore       26-Apr-83  AUTHOR
C#TYPE    Date utility
C#FORMAL PARAMETERS
Cin       rawdat  raw date in RELATE format
C                 Bits L1-12=year,R0-3=month,R4-8=day
C#CALLER  various
C#METHOD
C  zeros the unused bits: L0, L13-15, R9-15
C##
```

```
C     DDATE*************************************************
$CONTROL check=3
      INTEGER*4 FUNCTION ddate(dum)
      integer dum
C*                                         *** ABSTRACT ***
C#PURPOSE Returns the current date in RELATE I*4 format, which
C is year in bits 1-12 of left word, month in 0-3 of right word,
C day in 4-8 of right word, and all other bits unused.
C See the documentation on the data type DDATE in TDDATE.INCL.
C#AUDIT HISTORY
C          MSCarey        28-feb-83  AUTHOR
C          Densmore       26-Apr-83  Moved from DMUTIL to UTILA
C                                    and fixed LEAP YEAR part
C          Densmore       14-Oct-83  Changed lmonth data to function
C#FORMAL PARAMETERS
Cin       dum      dummy
C#COMMON BLOCKS
C         none
C#CALLER  various
C#METHOD
C  Calls calendar intrinsic, converts to month-day-year, and packs
C  output variable.
C#LOCAL VARIABLES
C         date     date as returned by intrinsic
C         year     year
C         days     day in year
C         td       a running total of days
C         month    month of year
C         dayom    day of month
C##
```

```
C     DDTOCD*****************************************************
$CONTROL check=3
      CHARACTER*10 FUNCTION ddtocd(ddate)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER*4 ddate
C*                                          *** ABSTRACT ***
C#PURPOSE Converts a DDATE into a string MM/DD/YYYY
C         See the include file TDDATE.INCL describing data type
C#AUDIT HISTORY
C         Densmore        26-Apr-83  AUTHOR
C#TYPE    Date utility
C#FORMAL PARAMETERS
Cin       ddate   a DDATE as described in TDDATE.INCL --
C                 it is in RELATE format: Bits L1-12=year
C                 Bits R0-3=Month, and Bits R4-8=Day
C#CALLER  various
C#METHOD
C  unpacks ddate and encodes string
C##
```

```
C     DDTOID*****************************************************
$CONTROL check=3
      SUBROUTINE ddtoid(ddate,month,day,year)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER*4 ddate
      INTEGER month,day,year
C*                                               *** ABSTRACT ***
C#PURPOSE Converts a DDATE into a string MM/DD/YYYY
C         See the include file TDDATE.INCL describing data type
C#AUDIT HISTORY
C         Densmore        26-Apr-83  AUTHOR
C#TYPE    Date utility
C#FORMAL PARAMETERS
Cin       ddate   a DDATE as described in TDDATE.INCL --
C                 it is in RELATE format: Bits L1-12=year
C                 Bits R0-3=Month, and Bits R4-8=Day
Cout      month   the integer month given in ddate [1..12]
Cout      day     the integer day [1..31]
Cout      year    the integer year (ie. 1983)
C#CALLER  various
C##
```

6u

```
C     DEARLY ***********************************************
$CONTROL check=3
       LOGICAL FUNCTION DEARLY (FRSTDAT,LASTDAT)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER*4 FRSTDAT,LASTDAT
C*                                          *** ABSTRACT ***
C#PURPOSE True if first date earlier than last date.  Clarifies.
C#AUDIT HISTORY
C         Mutchler        !d-mmm-yy  AUTHOR
C#FORMAL PARAMETERS
Cin      frstdat  first date
Cin      lastdat  second date
C#COMMON BLOCKS
C         NONE
C#CALLER  FLREPT AND BGREPT
C#METHOD
C  DOES A CLARIFY AND THEN USES TODATE ROUTINES
C#LOCAL VARIABLES
C        datfrst  flag
C##
```

```
C     DEBUG *************************************************
$CONTROL check=3,SEGMENT=utlr
      LOGICAL FUNCTION debug(idum)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      integer idum
C*                                           *** ABSTRACT ***
C#PURPOSE Checks lprnton job control word to see if in debug mode.
C#AUDIT HISTORY
C         MSCarey         28-jul-84  AUTHOR
C#TYPE    utility
C#FORMAL PARAMETERS
Cin       idum     dummy parameter to meet HP calling standards
C#COMMON BLOCKS
C         none
C#CALLER  various
C#METHOD
C  Use the findjcw system intrinsic to read the lprnton job
C  control word.  If it is 1, debug is true.
C#LOCAL VARIABLES
C         jcwnam  name of jcw to read
C##
```

```
C     DELIM**************************************************
$CONTROL check=2
      SUBROUTINE delim(string,first,last,length)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      PARAMETER m=1024
      CHARACTER*1 string(m)
      INTEGER first,last,length
C*                                          *** ABSTRACT ***
C#PURPOSE to discover the extent of a delimited text string
C#AUDIT HISTORY
C         Densmore        27-Oct-82  AUTHOR
C#TYPE    Simple subroutine
C#FORMAL PARAMETERS
Cin       string  delimited text string -- the text is delimited
C                 by a unique character occurring before and after.
C         first   location after first nonblank char in string
C         last    location before second occurrence of delimiter
C         length  the number of characters in the enclosed string
C#METHOD
C  Uses the HP Fortran byte addressing capability to locate
C  the first nonblank character, then the second occurrence
C  of the delimiter.  Note that the input string may be any
C  length up to m characters.
C##
```

```
C     DINDEX ***********************************************
$control check=2
      INTEGER FUNCTION dindex(string,lstr,substr,lsub)
      character*1 string(255),substr(255)
      integer lstr,lsub
C*                                          *** ABSTRACT ***
C#PURPOSE A version of HP function 'index' for delimed srch str
C#AUDIT HISTORY
C        MSCarey          09.may.83  AUTHOR
C#FORMAL PARAMETERS
Cin      string   possibly delimited string to look for substring in
Cin      substr   substring to look for in string
Cin      lstr     length of string
Cin      lsub     length of substring
C#COMMON BLOCKS
C     none
C#CALLER various
C#METHOD
C  Loop over string, looking for a match
C##
```

```
C     DPAUSE************************************************
$CONTROL check=3
      SUBROUTINE dpause(wait)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      real wait
C*                                          *** ABSTRACT ***
C#PURPOSE Causes process to pause wait seconds.
C#AUDIT HISTORY
C         MSCarey        10-dec-83  AUTHOR
C#FORMAL PARAMETERS
Cin       wait    number of seconds to pause
C#COMMON BLOCKS
C#CALLER  various
C#METHOD
C  Calls system intrinsic pause.
C#LOCAL VARIABLES
C         none
C##
```

```
C     EJECT*****************************************************
$CONTROL check=3
      SUBROUTINE eject(unit)
      INTEGER unit
C*                                            *** ABSTRACT ***
C#PURPOSE sends a page eject down to specified unit
C#AUDIT HISTORY
C         Densmore         21-Mar-83  AUTHOR
C#TYPE    screen utility
C#FORMAL PARAMETERS
Cin       unit    logical unit number
C#COMMON BLOCKS
Cin       tty     terminal parameters
C##
```

```
C     ELIMBL*************************************************
      SUBROUTINE ELIMBL (INSTR,LSTR,OUTSTR,LOSTR)
C*                           *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%include readc
      CHARACTER OUTSTR*(LSTR),INSTR*(LSTR)
      INTEGER LSTR,LOSTR
C*                                                  *** ABSTRACT ***
C#PURPOSE eliminate leading and trailing blanks from a string
C         and give its non-padded length
C#AUDIT HISTORY
C         MEMutchler      18 JAN 83  AUTHOR
C#TYPE    character string utility
C#FORMAL PARAMETERS
Cin       instr   text to be stripped of leading and trailing
C                 blanks
Cin       lstr    maximum length of strings
Cout      outstr  text stripped of leading and trailing blanks
Cout      lostr   length of outstr
C#COMMON BLOCKS
Cin       incpar  global parameter statement
C#METHOD
C  loop through string and count
C#LOCAL VARIABLES
C         string  temporary storage of text
C         blank   ' '
C##
```

```
C     ERRMSG*****************************************************
      SUBROUTINE ERRMSG (DELSTR)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
      CHARACTER DELSTR*LLINE
C*                                             *** ABSTRACT ***
C#PURPOSE Takes a delimited text string error message held
C          at delstr and outputs it to iout
C#AUDIT HISTORY
C          MEMutchler           18 JAN 83  AUTHCR
C#TYPE    utility
C#FORMAL PARAMETERS
Cin        delstr  delimited text string to be output
C#COMMON BLOCKS
Cin        incpar  global parameter statement
Cin        ioc     i/o file assignments
C#METHOD  Undelimit text string, get its lenght and write it.
C#LOCAL VARIABLES
C          output  undelimited text string
C          lenout  length of 'output' in non-blank characters
C##
```

```
C     FDDATE**************************************************
$CONTROL segment=asgnd,check=3
      INTEGER*4 FUNCTION fddate(ddatel,idurat)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER idurat
      INTEGER*4 ddatel
C*                                             *** ABSTRACT ***
C#PURPOSE Returns date of the first date in the period
C#AUDIT HISTORY
C         Densmore        17-Jun-83  AUTHOR
C#TYPE    date utility
C#FORMAL PARAMETERS
Cin       ddatel  date within the first period
Cin       idurat  duration (of each period) index
C            1=Fyear 2=Cyear 3=quarter 4=month 5=week 6=day
C#COMMON BLOCKS
Cin       tddate  ddate data type block
C#METHOD
C  Convert, push back to period's start, and convert back.
C##
```

```
C     FEXIST*************************************************
      LOGICAL FUNCTION fexist(filnam,len)
      integer len
      character*(len) filnam
C*                                                 *** ABSTRACT ***
C#PURPOSE Returns true if the given file already exists as a
C         permanent file.
C#AUDIT HISTORY
C        MSCarey          28-feb-83  AUTHOR
C#FORMAL PARAMETERS
Cin      len      length of string
Cin      filnam   name of file to check for, with extents if any
C#COMMON BLOCKS
C        none
C#CALLER various
C#METHOD
C  Calls intrinsic fopen specifying file as old.  If error indicating
C  it doesn't exists then fexist is false. Close file if open succeeds.
C  Test is on both job temporary file domain and perm file domain.
C#LOCAL VARIABLES
C        foptions bit-map argument for fopen
C        aoptions "
C        ifoptions"
C        iaoptions"
C        filnum   MPE file number returned by fopen
C        ercode   error code returned by fcheck
C        disp     file disposition argument for fclose
C        string   warning message buffer
C##
```

```
C     FILCLS**************************************************
      SUBROUTINE filcls(unit,ok,param)
      integer unit
      character*255 param
      logical ok
C*                                              *** ABSTRACT ***
C#PURPOSE Closes files opened by filopn.
C#AUDIT HISTORY
C        MSCarey          27-JAN-83  AUTHOR
C#FORMAL PARAMETERS
Cin       unit    fortran logical unit to be closed
Cin       param   delimited string holding control arguments separated
C                 by commas. Options are limited to SAVE,DELETE,MEONLY
Cout      ok      true if close successful
C#COMMON BLOCKS
Cio       untref  cross ref of MPE file number with fortran unit nums
C#CALLER  various
C#METHOD
C  Calls system intrinsic fclose.
C#LOCAL VARIABLES
C        dispos   transfers file disposal status
C        secode   transfers file security status
C        mesg     holds an error message
C        fnum     MPE file number
C        ercode   error code returned by fcheck
C##
```

```
C     FILOPN*************************************************
      SUBROUTINE filopn(unit,ok,param)
      integer unit
      logical ok
      character*255 param
C*                                              *** ABSTRACT ***
C#PURPOSE Opens HP files programmatically.
C#AUDIT HISTORY
C         MSCarey         30-JAN-83  AUTHOR
C#FORMAL PARAMETERS
Cin       unit    fortran logical unit number
Cout      ok      flag set to true if open successful; if false,
C                 likely cause is someone else having lock or
C                 sole access to desired file.  More serious errors
C                 cause abort calls from this routine.
Cin       param   delimited character string containing legal
C                 arguments separated by commas, as in
C                 ":NAME=JUNK,NEW,ASCI,FIXL,SEQL,RECL=128,NREC=1000:"
C#COMMON BLOCKS
Cout      untref  cross ref of MPE file nums & fortran logical units
C#CALLER  various
C#METHOD
C  Decodes arguments, checks for consistency, and calls MPE intrinsic
C  FOPEN and fortran library routine FSET.  See intrinsics manual,
C  Fortran manual section 8 for more on these.  String argument is
C  decoded into two arrays, one holding params and the other values
C  attached to the parameters where applicable.  For each, a list of
C  legal parameters is searched for a match.  The index of the match
C  is used as a reference by a computed goto to code setting parameters
C  for the fopen call.  Error checking is done after the FOPEN by a call
C  to intrinsic fcheck to identify conditions mandating an abort.
C#LOCAL VARIABLES
C        nparms  number of string parameters decoded
C        foption bit-mapped word for passage to FOPEN
C        lfoption logical of this word
C        aoption similar
C        laoption  "
C        toption   "
C        ltoption  "
C        mesg     error message
C        filz     file size as a double integer, required by FOPEN
C        fnum     MPE system file number
C        ercode   error code returned by fcheck
C        block    number of records per block
C        arg      array holding decoded alpha parameters
C        value    array holding values corresponding to args
C        option   array initialized to legal arg values
C        name     name of file to be opened
```

```
C         group    group user is currently logged onto
C         filsiz   maximum size of file in records (block if fixl)
C         nparms   number of character parameters found
C         recsiz   size of record in bytes
C         recpbl   number of records per block
C         igo      computed goto index
C         argnum   number of argument being processed by goto
C##
```

```
C     FINMEM ***********************************************
$CONTROL segment=seg'
      SUBROUTINE finmem(id,code)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer id,code
C*                                         *** ABSTRACT ***
C#PURPOSE   De-allocates an extended memory buffer.
C#AUDIT HISTORY
C        MSCarey        11-aug-83  AUTHOR
C#FORMAL PARAMETERS
Cin      id       operating system id code
Cin      code     id code supplied by user to inimem
C#COMMON BLOCKS
C#CALLER various
C#METHOD
C     Calls freedseg
C##
```

```
C     FLT***************************************************
      SUBROUTINE FLT ( BUFFER,LENBUF,ERROR,NUMBER )
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER LENBUF
       CHARACTER BUFFER *( LENBUF)
       REAL NUMBER
       LOGICAL ERROR
C*                                         *** ABSTRACT ***
C#PURPOSE  set number<-rnum(buffer), if possible
C          pointed to by IPTR
C#AUDIT HISTORY
C         MEMutchler          7  FEB 83  AUTHOR
C         MEMutchler         7 FEB     TESTER
C#TYPE    convert string to corresponding real value if possible
C#FORMAL PARAMETERS
Cin       buffer  string containing character version  of real
Cin       lenbuf   non-blank length of buffer
Cout      error    true iff buffer doesn't contain a real
Cout      number   real number found in buffer
C#COMMON BLOCKS  NONE
C#METHOD  determine if real number in string, else err = true
C#LOCAL VARIABLES
C         none
C##
```

```
C     GDATEP**************************************************
$CONTROL segment=asgnd,check=3
      INTEGER*4 FUNCTION gdatep(pern,idurat,fddate)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER pern,idurat
      INTEGER*4 fddate
C*                                                *** ABSTRACT ***
C#PURPOSE Returns the date of the first day in the pern'th period
C#AUDIT HISTORY
C         Densmore        17-Jun-83  AUTHOR
C#TYPE    date utility
C#FORMAL PARAMETERS
Cin       pern    period number to be converted to a date
Cin       idurat  duration (of each period) index
C            1=Fyear 2=Cyear 3=quarter 4=month 5=week 6=day
Cin       fddate  date of first day of first period
C#COMMON BLOCKS
Cin       tddate  data type for RELATE ddate
C#METHOD
C  convert, increment, convert
C##
```

```
C     GETGRP**************************************************
$CONTROL SEGMENT=SEG'
      SUBROUTINE GETGRP
C*                         *** FORMAL PARAMETER DECLARATIONS ***
C*                                                *** ABSTRACT ***
C#PURPOSE get group name and name's length in which runtime
C          menu system files should be located
C#AUDIT HISTORY
C          MEMutchler      10-MAR-83  AUTHOR
C#TYPE     menu system utility
C#FORMAL PARAMETERS  none
C#COMMON BLOCKS
Cout       envirn  holds info about  runtime enviorment
C#CALLER  ppinit and inimnu
C#METHOD
C          Inspect the proper Job Control Word flag's
C     value.  JCW should be absent or 0 unless the user
C     has given the DEVELOP UDC command.
C          Set variables used to determine which group
C     system files and relations are to be found in.
C#LOCAL VARIABLES
C          jcwnam   job control word name
C##
```

```
C     GETMEM ************************************************
$CONTROL check=2,segment=seg'
      SUBROUTINE getmem(id,length,source,start)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      integer id,length,start
      logical source(1)
C*                                           *** ABSTRACT ***
C#PURPOSE   Swaps data from extended memory into source array.
C#AUDIT HISTORY
C        MSCarey        11-aug-83  AUTHOR
C#FORMAL PARAMETERS
Cin      id       operating system id code for area
Cin      length   number of *2 words to swap
Cin      source   target array for words
Cin      start    starting position in extended mem to grab from
C#COMMON BLOCKS
C        none
C#CALLER various
C#METHOD
C     Calls dmovin.
C#LOCAL VARIABLES
C        lid      segment id
C##
```

```
C     GPERN*****************************************************
$CONTROL segment=asgnd,check=3
      INTEGER FUNCTION gpern(ddatel,numper,idurat,fddate)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER numper,idurat
      INTEGER*4 ddatel,fddate
C*                                               *** ABSTRACT ***
C#PURPOSE Returns period number given a RELATE ddate
C#AUDIT HISTORY
C         Densmore        17-Jun-83  AUTHOR
C#TYPE    date utility
C#FORMAL PARAMETERS
Cin       ddatel  the date to be converted
Cin       numper  maximum number of periods
Cin       idurat  duration (of each period) index
C            1=Fyear 2=Cyear 3=quarter 4=month 5=week 6=day
Cin       fddate  date of the first day in the first period
Cfunction gpern   returned period number, in [0..numper];
C                  0 is returned if ddatel is outside the range
C                  of valid period numbers
C#COMMON BLOCKS
Cin       tddate  RELATE ddate data type block
C#METHOD
C  Convert dates, take difference.  SEE SIMILAR CODE IN ASNLBS.
C#LOCAL VARIABLES
C         fm,fd,fy month/day/year of first period's first day
C         dm,dd,dy month/day/year corresponding to ddatel
C         fidate/lidate  first/last indexed period in absolute time
C##
```

```
C     IAND*******************************************************
$CONTROL check=3
      INTEGER FUNCTION iand(m,n)
      INTEGER m,n
C     bitwise ...; uses HP-FTN 16-bit exprs...Densmore 28 July 1983
      INTEGER jm,jn
      LOGICAL lm,ln
      EQUIVALENCE (jm,lm), (jn,ln)
C*ENDDEC
      jm = m
      jn = n
      ln = lm.AND.ln
      iand = jn
      RETURN
C     -------------------------------------------------------
      ENTRY ior(m,n)
      jm = m
      jn = n
      ln = lm.OR.ln
      iand = jn
      RETURN
      END
```

```
C     IDAYS*************************************************
$CONTROL check=3
      INTEGER FUNCTION idays(m1,d1,y1,m2,d2,y2)
C*                           *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER m1,d1,y1,m2,d2,y2
C*                                                  *** ABSTRACT ***
C#PURPOSE Returns the *2 number of days between two dates.
C#AUDIT HISTORY
C          Densmore         04-May-83  AUTHOR
C#TYPE     date utility
C#FORMAL PARAMETERS
Cin        m1,d1,y1 month/day/year of first date
Cin        m2,d2,y2 month/day/year of second date
C#METHOD  Subtracts mrkdays (which checks date validity);
C  See Function JDAYS for an INTEGER*4 version.
C##
```

```
C     IDTODD*************************************************
$CONTROL check=3
      SUBROUTINE idtodd(ddate,month,day,year)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER*4 ddate
      INTEGER month,day,year
C*                                         *** ABSTRACT ***
C#PURPOSE Converts to RELATE internal DDATE format; see TDDATE.INCL
C#AUDIT HISTORY
C        Densmore       11-May-83  AUTHOR
C#TYPE    Date Utility
C#FORMAL PARAMETERS
Cout     ddate   output date in RELATE format
Cin      month,day,year   input date in integer form
C##
```

```
C     INIIOC**************************************************
$CONTROL SEGMENT=MENU
      SUBROUTINE INIIOC
C*                                                *** ABSTRACT ***
C#PURPOSE initializes file assignments of i/o files found in
C         common /ioc/
C#AUDIT HISTORY
C         MEMutchler           18 JAN 83  AUTHOR
C         MEMutchler           8  FEB 83  TESTER
C#TYPE    mnugen and mnurun utility
C#COMMON BLOCKS
Cout      ioc     i/o file assignments
C#
```

```
C     INIMEM ************************************************
$CONTROL segment=seg'
      SUBROUTINE inimem(id,len,code,unique)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer id,code,len
      logical unique
C*                                          *** ABSTRACT ***
C#PURPOSE   Allocates an extended memory buffer.  On HP 3000
C     this means getting an extra data segment.
C#AUDIT HISTORY
C        MSCarey        11-aug-83  AUTHOR
C#FORMAL PARAMETERS
Cout     id       id number returned for use by putmem and getmem
Cin      code     id code provided by user to make unique segment
Cin      len      length of buffer desired in 2-byte word
Cout     unique   true if buffer requested did not already exist
C#COMMON BLOCKS
C!       !        !
C#CALLER various
C#METHOD
C     Call to getdseg.
C##
```

```
C     INIPRC *************************************************
$CONTROL segment=seg'
      SUBROUTINE iniprc
C*                      *** FORMAL PARAMETER DECLARATIONS ***
C*                                            *** ABSTRACT ***
C#PURPOSE   INItialize module PRoCess.  Does all necessary
C initialization for a module son process.
C#AUDIT HISTORY
C         MSCarey        28-jun-83  AUTHOR
C#FORMAL PARAMETERS
C         none
C#COMMON BLOCKS
Cout      pvalue   menu system parameter values
C#CALLER various
C#METHOD
C     Calls to other initialization routines. Swaps in /pvalues/
C     /scenar/,/lprnts/,/ioc/,/io/ from an extra data segment
C#LOCAL VARIABLES
C         name     file name lccref.mnurel/makmenu
C##
```

```
C     INITIO*************************************************
      SUBROUTINE initio
C
C  Initializes the most necessary I/O unit numbers
C  Note that this routine does NOT use include directives
C  so that the utility library need not be compiled with INCL
C#
```

```
C      IXSUM***************************************************
$CONTROL check=3
      INTEGER FUNCTION ixsum(n,v)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER n
      INTEGER v(n)
C*                                             *** ABSTRACT ***
C#PURPOSE Sums the cross section vector V -- integer sum
C#AUDIT HISTORY
C          Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       n       length of v
Cin       V       vector of integers whose elements to sum
C                 ____n
Cfunction ixsum   C                 /___i=1   i
C#COMMON BLOCKS
C      none
C##
```

```
C     JDAYS*****************************************************
$CONTROL check=3
      INTEGER*4 FUNCTION jdays(m1,d1,y1,m2,d2,y2)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER m1,d1,y1,m2,d2,y2
C*                                         *** ABSTRACT ***
C#PURPOSE Returns *4 number of days between two dates.
C#AUDIT HISTORY
C         Densmore        26-May-83  AUTHOR
C#TYPE    date utility
C#FORMAL PARAMETERS
Cin       m1,d1,y1 month/day/year of first date
Cin       m2,d2,y2 month/day/year of second date
C#METHOD  Subtracts mrkdays (which checks validity);
C  See Function IDAYS for an INTEGER*2 version.
C##
```

```
C     JHASH**************************************************
$CONTROL check=3
      SUBROUTINE jhash(a,kmax,nrec,k,amin,amax,ih,nh)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER kmax,nrec,k,nh, ih(nh)
      INTEGER*4 a(kmax,nrec), amin, amax
C*                                          *** ABSTRACT ***
C#PURPOSE Returns the sorted order of the I*4 records A, based on row k
C#AUDIT HISTORY
C          Densmore       16-Jun-83  AUTHOR
C#TYPE    Sort utility
C#FORMAL PARAMETERS
Cin       a        the array of nrec records, each of length kmax
Cin       kmax     the length of each record
Cin       nrec     the number of records
Cin       k        the element of each record on which to sort
Cin       amin     a lower bound on the values a(k,*)
Cin       amax     an upper bound on the values a(k,*)
Cout      ih       the sorted order of the records contained in a,
C                  based on the element k in each record.  That is,
C                  ih(1) contains the number of the record which
C                  appears first when they are given in order; ih(2)
C                  contains the number of the second record, etc.
C         nh       the length of the array ih.  This number must
C                  obviously be >= nrec; ih is actually used as a
C                  work area and should be at least 2*nrec, preferably
C                  3*nrec.
C#METHOD
C  Assumes that the records are approximately linearly distributed.
C  Takes the value of each record's key and uses it to estimate its
C  sequence number, placing that record's index number in the ih
C  element corresponding to that sequence number.  This is repeated
C  for each record, resolving collisions as required.  If only a
C  few collisions need to be resolved this is a nearly linear order-
C  ing algorithm.  At the end the nonzero (unfilled) elements of the
C  ih array are removed and the filled elements left shifted so that
C  the first nrec elements of ih give the ordering information.
C##
```

```
C     KFIX***********************************************
      SUBROUTINE KFIX ( BUFFER,LENBUF,ERROR,NUMBER )
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER LENBUF
       CHARACTER BUFFER *(LENBUF)
       INTEGER NUMBER
       LOGICAL ERROR
C*                                          *** ABSTRACT ***
C#PURPOSE  set number<-inum(buffer), if possible
C          pointed to by IPTR
C#AUDIT HISTORY
C         MEMutchler           7  FEB 83  AUTHOR
C         MEMutchler          7 FEB     TESTER
C#TYPE    convert string to corresponding integer value if possible
C#FORMAL PARAMETERS
Cin       buffer  string containing character version  of integer
Cin       lenbuf   non-blank length of buffer
Cout      error    true iff buffer doesn't contain a integer
Cout      number   integer number found in buffer
C#COMMON BLOCKS  NONE
C#METHOD  determine if integer number in string, else err = true
C#LOCAL VARIABLES
C         none
C##
```

```
C     LABORT*****************************************************
      SUBROUTINE LABORT (INTVAR, STRING)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
      CHARACTER STRING*LLINE
      INTEGER INTVAR
C*                                               *** ABSTPACT ***
C#PURPOSE causes a program abort and writes and diagnostic message
C         of "intvar:  string".
C#AUDIT HISTORY
C         MEMutchler          18 JAN 83  AUTHOR
C#TYPE    abort and message
C#FORMAL PARAMETERS
Cin       intvar  integer variable to be in diagnostic
Cin       string  delimited text string to be in diagnostic
C#COMMON BLOCKS
Cin       incpar  global parmeter statements
C#METHOD
C  concatenate to get dts string to output
C#LOCAL VARIABLES
C         nstring   undelimited output string
C         lenstr    string length in non-blank characters
C         lout      length of input message string, undelimited
C         buffer    delimited output string
C##
```

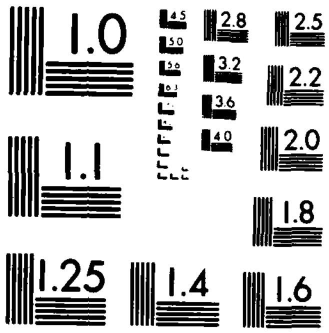
```
C     LATDAT**********************************************************
$CONTROL CHECK=3
      SUBROUTINE LATDAT (DATE)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER*4 DATE
C*                                                  *** ABSTRACT ***
C#PURPOSE get maximum date value
C#AUDIT HISTORY
C          MEMutchler      31 may 83  AUTHOR
C#TYPE     relate date utility
C#FORMAL PARAMETERS
Cou        date     maximum date value
C#METHOD
C  set date to greatest *4 value and clarify
C##
```

```
C     LBIT*****************************************************
$CONTROL check=3
      LOGICAL FUNCTION lbit(word,pos)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER word,pos
C*                                            *** ABSTRACT ***
C#PURPOSE returns whether bit "pos" of "word" is set
C#AUDIT HISTORY
C          Densmore        01-Apr-83  AUTHOR
C#TYPE     general utility
C#FORMAL PARAMETERS
Cin        word    a sixteen-bit word, of which only first 15 are used
Cin        pos     bit position; [1..15]
C#METHOD
C  Note that the bits are numbered from right to left.
C##
```

```
C     LETNUM*************************************************
$control check=2
      LOGICAL FUNCTION LETNUM(string,len)
      integer len
      character*255 string
C*                                          *** ABSTRACT ***
C#PURPOSE checks a string for characters other than letters or
C          numbers.  True if no such characters.
C#AUDIT HISTORY
C        MSCarey          28-feb-83  AUTHOR
C#FORMAL PARAMETERS
Cin      len      length of input string
Cin      string   string to check
C#COMMON BLOCKS
C        none
C#CALLER  various
C#METHOD
C  Looks for characters outside permitted octal code ranges
C##
```

AD-A150 423 ALIAS (ACQUISITION AND LOGISTICS INFORMATION AND ANALYSIS SYSTEM) MAINTEN..(U) DECISION-SCIENCE APPLICATIONS INC ARLINGTON VA M S CAREY ET AL. 2/7

UNCLASSIFIED 31 OCT 84 DSA-593-VOL-2 N00014-82-C-0813 F/G 15/5 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
C     LETONL ***********************************************
$control segment=dmaint
      LOGICAL FUNCTION letonl(string,len)
      integer len
      character*(len) string
C*                                                *** ABSTRACT ***
C#PURPOSE Checks a string to be sure t contains only letters or " ".
C#AUDIT HISTORY
C        MSCarey          17-mar-83  AUTHOR
C#FORMAL PARAMETERS
Cin      len      length of input string
Cin      string   string to be checked
C#COMMON LOCKS
C        none
C#CALLER various
C#METHOD
C  Checks to make sure each byte is within the proper octal range.
C##
```

```
C     LISTON ************************************************
$CONTROL segment=seg'
      SUBROUTINE liston(scenar,mnunam,list,lchars,mxnlst,numon,tomany)
      character*(lchars) list(mxnlst)
      character*12 scenar,mnunam*8
      integer lchars,mxnlst,numon
      logical tomany
C*                                            *** ABSTRACT ***
C#PURPOSE   Reads a list menu relation and returns a list
C           of candidates which are "on".
C#AUDIT HISTORY
C        MSCarey          02-Jun-83  AUTHOR
C        Densmore         10-Jun-83  Added neglected 'tomany' formal
C#FORMAL PARAMETERS
Cin      scenar   current scenario name
Cin      mnunam   list menu for which list is desired
Cin      lchars   max chars in a list candidate
Cin      mxnlst   max number of candidates returnable
Cout     list     list of candidates which are "on"
Cout     numon    number of candidates returned
Cout     tomany   more found on than allowed by mxnlst
C#COMMON BLOCKS
Cin      prmcrs   permanently open cursor indexes
Cin      envrn    group name for list relations
Cio      rcrd01   buffer for list retrievals
C#CALLER various
C#METHOD
C     Look in the cross reference relation for the relation name
C     holding candidate statuses for the given menu.  Open that
C     relation.
C     Calc to the first tuple for the given scenario
C     Read sequentially until all tuples for that scenario are
C     found, placing the candidate field for each on the list
C     if its status is "on".  Close the relation and return.
C#LOCAL VARIABLES
C        cand     candidate name
C        stat     candidate status
C##
```

```
C     LMONTH**************************************************
$CONTROL check=3
      INTEGER FUNCTION lmonth(month,year)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER month,year
C*                                             *** ABSTRACT ***
C#PURPOSE Returns number of days in given month
C#AUDIT HISTORY
C          Densmore        11-Oct-83  AUTHOR
C#TYPE    Date utility
C#FORMAL PARAMETERS
Cin       month   integer representation of month
Cin       year    integer year (e.g. 1983)
C#METHOD
C  uses array indexed by month; feb is special case.
C##
```

```
C     LPSEND ***********************************************
$CONTROL check=3,segment=devctrl
      SUBROUTINE lpsend(unit)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer unit
C*                                           *** ABSTRACT ***
C#PURPOSE Closes printer spool file, causing output to commence.
C#AUDIT HISTORY
C         MSCarey         20-sep-83  AUTHOR
C#TYPE    utility
C#FORMAL PARAMETERS
Cin       unit     unit number lp file is open on
C#CALLER  various
C#METHOD
C  Closes file unless it is $stdlist (terminal).
C##
```

```
C     LPSET*************************************************
$CONTROL segment=devctrl,check=3
      SUBROUTINE lpset(unit)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER unit
C*                                           *** ABSTRACT ***
C#PURPOSE Determines Line Printer UNIT number
C         and opens spool file if appropriate
C#AUDIT HISTORY
C         Densmore        26-May-83  AUTHOR
C         MSCarey         29-Jun-83  'undummied' to use PVALUE
C#TYPE    I/O Utility
C#FORMAL PARAMETERS
Cout      unit  unit to use for printer output
C#COMMON BLOCKS
Cout      ioc     also changes lp unit in this common
C#CALLER  anyone who wants to write to a line printer
C##
```

```
C     LSTRNG******mstrng**************************************
$CONTROL check=2
      SUBROUTINE lstrng(sin,lin0,sout,lout,mlout)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER lin0,lout,mlout
      CHARACTER*1 sin(1024), sout(mlout)
C*                                           *** ABSTRACT ***
C#PURPOSE moves possibly delimited string sin to sout
C#AUDIT HISTORY
C         Densmore        15-Dec-82  AUTHOR
C#TYPE    string manipulation utility
C#FORMAL PARAMETERS
Cin       sin     input string
Cin       lin0    length of sin; if lin0=0, then sin is a DTS
Cout      sout    output string
Cout      lout    actual length of sout
Cin       mlout   maximum allowable length of sout
C#METHOD
C  Determines first,last, and length; uses HP Fortran character
C  assignment with substring operators.  Note that sout may share
C  addresses with sin, since the assignment operations are
C  buffered.
C
C##
```

```
C     LTRIM**************************************************
$CONTROL check=2
      INTEGER FUNCTION ltrim(string,len)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER len
      CHARACTER*1 string(len)
C*                                          *** ABSTRACT ***
C#PURPOSE returns position of first nonblank character in string
C#AUDIT HISTORY
C         Densmore        20-Jan-83  AUTHOR
C#TYPE    character utility
C#FORMAL PARAMETERS
Cin       string  character string
Cin       len     length of string
C##
```

```
C     LWARN****************************(*****************(*
      SUBROUTINE LWARN (INTVAR, STRING)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
      INTEGER INTVAR
      CHARACTER STRING*LLINE
C*                                              *(* ABSTRACT ***
C#PURPOSE causes a program warning and writes and diagnostic message
C         of "intvar;  string".
C#AUDIT HISTORY
C         MEMutchler            18 JAN 83  AUTHOR
C#TYPE    mnugen utility
C#FORMAL PARAMETERS
Cin       intvar  integer to go into diagnostic message
Cin       string  char. string to go into diagnostic
C#COMMON BLOCKS
Cin       ioc     i/o assignments
C#LOCAL VARIABLES
C         nstring undelimited input string
C##
```

```
C      MABORT**************************************************
       SUBROUTINE mabort(text)
C*                          *** FORMAL PARAMETER DECLARATIONS ***
       PARAMETER m=255
       CHARACTER text*m
C*                                                *** ABSTRACT ***
C#PURPOSE Print message to whatever units are appropriate, then abort
C#AUDIT HISTORY
C          Densmore        27-Oct-82  AUTHOR
C#TYPE     Simple subroutine (no output)
C#FORMAL PARAMETERS
Cin        text    delimited text string giving caller
C                  and an indication of the error that
C                  occurred.
C##
```

```
C     MATCH2*************************************************
$CONTROL check=2
      INTEGER FUNCTION match2(list,length,ientry)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER length,ientry
      INTEGER list(length)
C*                                             *** ABSTRACT ***
C#PURPOSE Makes first match of entry to list; returns index
C#AUDIT HISTORY
C         Densmore        08-Jun-83  AUTHOR
C#TYPE    data checking utility
C#FORMAL PARAMETERS
Cin       list    list of integer*2 items
Cin       length  length of list
Cin       ientry   item to check against list
C#METHOD
C  Simple do-loop
C##
```

```
C     MATCHC**************************************************
$control check=2
      INTEGER FUNCTION MATCHC(CHARAR,LENCH,LENARR,MATCH)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER LENCH,LENARR
      CHARACTER*(LENCH) MATCH,CHARAR(LENARR)
C*                                                  *** ABSTRACT ***
C#PURPOSE get index of match in charar array of character
C         strings
C#AUDIT HISTORY
C         MEMutchler      28-may-83  AUTHOR
C         Densmore        29-Jun-83   Moved to CUTILS
C#TYPE    character utility
C#FORMAL PARAMETERS
Cin       charar  array of strings to match into
Cin       lench   length of character strings
Cin       lenarr  length of array
Cin       match   string to match
C#METHOD
C  find match to array, match = position, 0 if not found
C##
```

```
C      MODCOR**************************************************
$CONTROL check=3
      INTEGER FUNCTION modcor(number,base)
      INTEGER number,base
C#PURPOSE Provide correct modulo function which is always
C  positive (0..base-1) instead of negative when number is.
C#AUTHOR  Densmore
C##
```

```
C     MONCOM***************************************************
      SUBROUTINE moncom(comand,succes)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      PARAMETER m=255, n=160
      CHARACTER comand*m
      logical succes
C*                                          *** ABSTRACT ***
C#PURPOSE to execute the string "comand" as a monitor command.
C#AUDIT HISTORY
C         Densmore        27-Oct-82  AUTHOR
C#TYPE    Simple subroutine
C#FORMAL PARAMETERS
Cin       comand  delimited text string giving command to
C                 be executed
Cout      succes  logical variable indicating if the command
C                 was successfully executed.
C#METHOD
C  Calls delim to determine the extent of the actual command
C  text, then places it in a buffer (maximum length of a command
C  is n-1 = 159 characters).  A carriage return is appended
C  using the % construct valid for HP Fortran, then the command
C  instrinsic is called.  Success is given by the Condition Code
C  contruct .CC. and the ierr value, which is zero if okay.
C##
```

```
C     MPDCOD***************************************************
      SUBROUTINE mpdcod(param,arg,value,nparms,maxprm,lenprm)
      character*255 param
      character*(lenprm) arg(maxprm),value(maxprm)
      integer maxprm,nparms
C*                                          *** ABSTRACT ***
C#PURPOSE Takes a delimited string of parameters separated by
C         commas and decodes it into individual parameters,
C         also decoding individual parameters into left and
C         right sides of any embedded equal signs.
C#AUDIT HISTORY
C         MSCarey         30-JAN-83  AUTHOR
C#FORMAL PARAMETERS
Cin      param    delimited string to be decoded
Cout     arg      individual parameters, left side of equal sign
Cout     value    individual parameters, right side of equal sign
Cout     nparms   number of parameters found in decoding
Cin      maxprm   maximum number of parameters to decode
Cin      lenprm   maximum length of an arg or value after decoding
C#COMMON BLOCKS
C        none
C#CALLER  filopn,filcls
C#METHOD
C  Force to uppercase and un-delimit.  Then loop over number of commas
C  found, searching also for equal signs.  Blank the work array as
C  search moves to the right.
C#LOCAL VARIABLES
C        i,find,lind,len,iword,icom,ieq,lenv,lenp: char position
C                 or loop indexes
C        work     storage for decode of character parameter
C##
```

```
C       MRKDAY*********************************************
$CONTROL check=3
        INTEGER*4 FUNCTION mrkday(imonth,iday,iyear)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
        INTEGER imonth,iday,iyear
C*                                           *** ABSTRACT ***
C#PURPOSE Marks the day; returns #dys since a given date (e.g.
C        31-December-1600).  This routine is only required to
C        return relative values; the above date need not be
C        used, but subtracting two mark-days should yield the
C        number of days between the two corresponding dates.
C#AUDIT HISTORY
C        Densmore        04-May-83  AUTHOR
C#TYPE   Date utility
C#FORMAL PARAMETERS
Cin      imonth  integer representation of month [1..12]
Cin      iday    day [1..31]
Cin      iyear   year [1601..2399]
C#CALLER  various
C#RELATED ROUTINES
C  Other (self-contained) functions may depend on the actual
C  date used (31-Dec1600) to return other information.  For
C  example, NUMDAY returns [1..7] (ie. [Sun..Sat]) given a
C  date; this depends on the fact that 31-Dec-1600 was a Sunday.
C  Such routines, if they exist on this library, are NUMDAY and
C  DATEMK (which is the inverse of MRKAY).
C#METHOD
C  Checks that date is valid.  Determines number of full year days.
C  Determines number of full month days.  Adds leap year days.
C  Conditionally subtracts this leap year day.  Conditionally
C  subtrats Century non-leap year days.  Conditionally adds the
C  year 2000 leap year day.
C#LOCAL VARIABLES
C        idcum   number of days in a year to that month
C##
```

```
C     MTCHOC*************************************************
$CONTROL check=3
      INTEGER FUNCTION mtchoc(clist,nchar,len,ientry)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER nchar,len
      CHARACTER*(nchar) clist(len), ientry
C*                                          *** ABSTRACT ***
C#PURPOSE Tries to match entry to a list element, but for ordered list
C         (use matchc for unordered list)
C#AUDIT HISTORY
C         Densmore        10-Jun-83  AUTHOR
C#TYPE    character match utility
C#FORMAL PARAMETERS
Cin       clist   character array of items to match against
Cin       nchar   number of characters in each clist item
Cin       len     number of clist items
Cin       ientry   item against which to match
C#CALLER  utility
C#METHOD
C  Binary search...returns 0 if no match exists
C##
```

```
C     NCFW********nwfc********etc.**************************
      INTEGER FUNCTION ncfw(nwords)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER nwords,nchars
C*                                               *** ABSTRACT ***
C#PURPOSE convert from word sizes to character sizes & vice versa
C#AUDIT HISTORY
C         Densmore        15-Dec-82  AUTHOR
C#TYPE    character manipulation utility
C#FORMAL PARAMETERS
Cin       nwords  number of words
Cin       nchars  number of characters
C#MNEMONICS
C         N = number                C = Characters
C         F = From                  W = Words
C         SW= ShortWords (*2)       LW= LongWords (*8)
C
C#ENTRIES                                        DESCRIPTION
C     ncfw (nwds4)  ::= 4*nwds4        *4 words to characters
C     ncfsw(nwds2)  ::= 2*nwds2        *2 words to characters
C     ncflw(nwds8)  ::= 8*nwds8        *8 words to characters
C     nwfc (nchars) ::= (nchars+3)/4  characters to *4 words
C     nswfc(nchars) ::= (nchars+1)/2  characters to *2 words
C     nlwfc(nchars) ::= (nchars+7)/8  characters to *8 words
C##
```

```
C     NUMASK************************************************
$CONTROL check=2
      SUBROUTINE numask(number,nchar,cmask,cout)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER number,nchar
      CHARACTER*(nchar) cmask,cout
C*                                          *** ABSTRACT ***
C#PURPOSE Uses cmask as a mask over which significant digits
C         in number are placed.
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       number  the value to use in overwriting mask
Cin       nchar   length of cmask and cout (result)
Cin       cmask   the character mask
Cout      cout    result
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#METHOD
C  Examples using notation making numask look like char*(*) function:
C     "00234" = numask(   234,5,"00000") | "    " = numask(  0,4,"    ")
C     "|3"    = numask(     3,2,"||")    | "####" = numask(  0,4,"####")
C     "*****" = numask(100000,5,"abcde") | "0-12" = numask(-12,4,"0000")
C##
```

```
C     NUMSFX*************************************************
$CONTROL check=3
      CHARACTER*2 function numsfx(number,ncaps)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER number,ncaps
C*                                           *** ABSTRACT ***
C#PURPOSE returns a number suffix, like "th" -- the "5th" item...
C#AUDIT HISTORY
C         Densmore        30-Mar-83  AUTHOR
C#TYPE    utility
C#FORMAL PARAMETERS
Cin       number  number for which suffix is to be provided
Cin       ncaps   set to 1 for lowercase, 2 for UPPERCASE
C#LOCAL VARIABLES
C         tenprt  "ten part" -- 10*(tens-digit) + (ones-digit)
C         oneprt  "one part" -- value of (ones-digit)
C##
```

```
C     NWDATE*********************************************
$CONTROL check=3
      INTEGER*4 FUNCTION nwdate(oldate,ndays)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER*4 oldate
      INTEGER ndays
C*                                          *** ABSTRACT ***
C#PURPOSE Returns the date ndays away from oldate
C#AUDIT HISTORY
C         Densmore        31-May-83  AUTHOR
C#TYPE    date utility
C#FORMAL PARAMETERS
Cin       oldate  old date...in RELATE format (see /TDDATE/)
Cin       ndays   number of days...may be positive or negative
C#CALLER  utility
C#METHOD
Converts to mm/dd/yy representation and uses datemk/mrkday
C*                     *** INCLUDES and LOCAL DECLARATIONS ***
C##
```

```
C     NWDATU*************************************************
$CONTROL check=2
      INTEGER*4 FUNCTION nwdatu(ddate,nper,pertyp)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER*4 ddate
      INTEGER nper
      CHARACTER*4 pertyp
C*                                          *** ABSTRACT ***
C#PURPOSE Adds given number periods to date
C#AUDIT HISTORY
C         Densmore        12-Oct-83  AUTHOR
C#TYPE    Date utility
C#FORMAL PARAMETERS
Cin       ddate   a RELATE date (not necessarily clarified)
Cin       nper    number of periods to add (+ or -)
Cin       pertyp  period type; may be 'DAY' 'WEE' 'MON' 'QUA' 'YEA'
C#METHOD
C  looks for which type; performs addition; checks that the
C  resulting date is still valid.
C#LOCAL VARIABLES
C         type    3-char version of pertyp
C         m,d,y   new date
C         inper   internal version of nper
C         movm,movy amounts to change month and year
C##
```

```
C     NWIDAT*************************************************
$CONTROL check=3
      SUBROUTINE nwidat(om,od,oy,ndays,nm,nd,ny)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER om,od,oy,ndays,nm,nd,ny
C*                                           *** ABSTRACT ***
C#PURPOSE Returns mm/dd/yy ndays away from input date
C#AUDIT HISTORY
C         Densmore        31-May-83  AUTHOR
C#TYPE    Date utility
C#FORMAL PARAMETERS
Cin      om/od/oy   old month/day/year
Cin      ndays      number of days separating old from new
Cout     nm/nd/ny   new month/day/year output
C#METHOD
C  Uses mrkday and datemk routines
C##
```

```
C     PGINIT ***********************************************
$CONTROL segment=pgprnt
      SUBROUTINE pginit
C*                          *** FORMAL PARAMETER DECLARATIONS ***
C*                                                *** ABSTRACT ***
C#PURPOSE   INITializes the PaGe printing subsystem.
C#AUDIT HISTORY
C         MSCAREY         05-sep-83  AUTHOR
C#FORMAL PARAMETERS
C         none
C#COMMON BLOCKS
Cout      pgsys    page printing utility control info
C#CALLER utility
C#METHOD
C     Opens the buffer file for the printer and does an intitial
C     reset of the buffer to empty.
C#LOCAL VARIABLES
C         ok        filopn flag
C##
```

```
C     PGRSET ************************************************
$CONTROL segment=pgprnt
      SUBROUTINE pgrset(unit,linlen,paglen,mode,fmode,quit,qchar)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      logical quit
      integer unit,linlen,paglen,mode,fmode
      character*1 qchar
C*                                           *** ABSTRACT ***
C#PURPOSE   ReSETs PaGe printing utility.
C#AUDIT HISTORY
C        MSCarey        05-sep-83  AUTHOR
C#FORMAL PARAMETERS
Cin      unit     unit number to send output to now
Cin      linlen   length of output lines
Cin      paglen   number of lines on a page now
Cin      mode     operating mode (see /pgsys/)
Cin      fmode    page feed mode
Cin      quit     true if user wants pg to prompt for quit
Cin      qchar    character to accept as quit signal
C#COMMON BLOCKS
Cio      pgsys    page printer globals
C#CALLER various
C#METHOD
C     Set up the common block variables according to the arguments
C#LOCAL VARIABLES
C        none
C##
```

```
C     PGSEND ***********************************************
$CONTROL segment=pgprnt
      SUBROUTINE pgsend(header,nchedr,hlines,line,eoblok,eopage,quit)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
%include pgsys
      integer nchedr,hlines
      character*(nchedr) header(hlines),line
      logical eoblok,eopage,quit
C*                                               *** ABSTRACT ***
C#PURPOSE   Accepts a line of output and places it in an output
C     buffer for eventual full-page printing.  Optionally,
C     controls the full-page output event and prompts the user
C     for his desire to page next.
C#AUDIT HISTORY
C        MSCarey        05-sep-83  AUTHOR
C#FORMAL PARAMETERS
Cin      header   page heading text
Cin      nchedr   number of characters in a header line
Cin      hlines   number of lines of heading text
Cin      line     line to be output
Cin      eoblok   true if line is end of a block which must fit on
Cout     eopage   true if page is now full
C#COMMON BLOCKS
Cio      pgsys    page printing utility globals
Cin      ioc      global io units
C#CALLER various
C#METHOD
C     Send the text to the output buffer (a file).
C     Jump to the code handling the current mode.
C     Prompt and/or call pgwrit to do the output and/or set eopage.
C#LOCAL VARIABLES
C        prompt   prompt string
C##
```

```
C     PGWRIT ***********************************************
$CONTROL segment=pgprnt
      SUBROUTINE pgwrit(header,nchedr,hlines,page)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
%include pgsys
      integer nchedr,hlines
      character header*(nchedr)(hlines)
      logical page
C*                                          *** ABSTRACT ***
C#PURPOSE   Writes out a page or line from the buffer and does
C     some buffer housekeeping
C#AUDIT HISTORY
C         MSCarey        05-sep-83  AUTHOR
C#FORMAL PARAMETERS
Cin       header   text of page header
Cin       nchedr    number of chars in header line
Cin       hlines   number of lines in header
Cin       page     true if in header to be written on each call
C                  if false, header written only on first call
C#COMMON BLOCKS
Cio       pgsys    page utility globals
C#CALLER various, mostly pgsend
C#METHOD
C     Write from pgtop to pglast; reset pgtop to record after pglast
C     or to 0 if this is > pgatln.  Write header according to mode.
C#LOCAL VARIABLES
C         line     line buffer for transfer from buffer file to output
C                  device
C##
```

```
C     PLURAL*************************************************
$CONTROL check=3
      CHARACTER*1 FUNCTION plural(number,case)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER number,case
C*                                         *** ABSTRACT ***
C#PURPOSE Returns "S" if number <> 1, returns blank if =1
C#AUDIT HISTORY
C         Densmore        21-Apr-83  AUTHOR
C#TYPE    Format utility
C#FORMAL PARAMETERS
Cin       number  value
Cin       case    1=lower case, 2=upper case
C##
```

```
C     PRTHLP*************************************************
$CONTROL check=3
      SUBROUTINE prthlp(name,found,in,out)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*8 name
      LOGICAL found
      INTEGER in,out
C*                                              *** ABSTRACT ***
C#PURPOSE prints the text in file UNIT associated with category NAME
C#AUDIT HISTORY
C         Densmore        29-Mar-83  AUTHOR
C#TYPE    utility for use with assigner
C#FORMAL PARAMETERS
Cin       name    char*8 name of category
Cout      found   .TRUE. if category name was found and printed
Cin       in      unit number for file on which text is located
Cin       out     display unit number
C#METHOD
C  The file associated with unit number IN is expected to have leader
C  lines associated with each category it contains.  These leader
C  lines are of the form:
C  %BEGIN CAT-NAME
C  where the first seven characters are "%BEGIN ", and the next eight
C  (8) characters are the category name.  Remaining characters on
C  these leader lines are ignored and may be used for comments.
C
C  When the names (ignoring case) match, the corresponding text
C  is printed until another %BEGIN, or End-Of-File, is encountered.
C  If the text found contains lines whose first seven characters are
C  "%BREAK ", at each such point the process is halted and the file
C  5 is queried for a carriage return to continue, except when
C  OUT is not file 6.
C##
```

```
C     PUTMEM ***********************************************
$CONTROL check=2,segment=seg'
      SUBROUTINE putmem(id,length,source,start)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      integer id,length,start
      logical source(1)
C*                                         *** ABSTRACT ***
C#PURPOSE   Swaps data in source into extended memory area.
C#AUDIT HISTORY
C        MSCarey        11-aug-83  AUTHOR
C#FORMAL PARAMETERS
Cin      id       operating system id code for area
Cin      length   number of *2 words to swap
Cin      source   source array for words
Cin      start    starting position in extended mem to send to
C#COMMON BLOCKS
C        none
C#CALLER various
C#METHOD
C     Calls dmovout.
C##
```

```
C     QSORTC**************************************************
$CONTROL check=2
      SUBROUTINE qsortc(a,n,c,s,l)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER n,c,s,l
      CHARACTER*(c) a(n)
C*                                            *** ABSTRACT ***
C#PURPOSE Quick SORTing method for Character arrays; uses HEAPSORT
C#AUDIT HISTORY
C         Densmore        17-May-83  AUTHOR
C#TYPE    Sort utility
C#FORMAL PARAMETERS
Cin/out   a       array to be sorted (in place)
Cin       n       number of elements in the array a
Cin       c       number of characters in each element of a
Cin       s       starting character of the key for each element
Cin       l       length of the substring comprising the key
C#METHOD
C  Uses HEAPSORT...See Knuth Volume 3 pp. 146-147
C
C  HeapSort is guaranteed to be an N*Log(N) algorithm even in worst
C  cases.  Records considered are those between i and j at any one
C  point in the algorithm.  If left>1, then a "Heap" is being formed,
C  such that a(floor(j/2))[s:l] > a(j)[s:l] for all j; j such that
C  1 <= floor(j/2) < j <= n.  Once left=1, a(1) has the largest
C  remaining key, and in this manner the records are sifted into a
C  sorted order, in place.
C##
```

```
C     QUERY**************************************************
$CONTROL check=2
      LOGICAL FUNCTION query(text)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*255 text
C*                                           *** ABSTRACT ***
C#PURPOSE prints query and calls yesno
C#AUDIT HISTORY
C         Densmore        10-Feb-83  AUTHOR
C#TYPE    I/O utility
C#FORMAL PARAMETERS
Cin       text    query text -- string is delimited
C##
```

```
C     RANF*****************************************************
$CONTROL check=3
      REAL FUNCTION ranf(iseq)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER iseq
      PARAMETER len1=11
      INTEGER*4 jseed(len1),jseed1
C*                                                 *** ABSTRACT ***
C#PURPOSE Generates uniform random numbers over the range (0,1)
C#AUDIT HISTORY
C         Densmore         16-Jun-83  AUTHOR
C#TYPE    statistics utility
C#FORMAL PARAMETERS
Cin       iseq    sequence number (0..len1-1 -- if not in this range
C                 then the 0 sequence is used)
Cin or out jseed  the seeds for each sequence
Cin       jseed1  a single seed which is used to init all seeds
C#METHOD
C  ranf returns a uniform random number on sequence iseq over (0,1)
C  ranset initializes all len1 sequences independently
C  ranstl initializes all seeds from a single input seed
C  ranget retrieves all len1 seeds for storage
C  ALL entries are functions because ranf is; only ranf uses the
C  function return.
C##
```

```
C     RDATE*************************************************
      REAL FUNCTION rdate(dum)
      integer dum
C*                                              *** ABSTRACT ***
C*PURPOSE Returns the current date as YYMMDD in a real variable.
C*AUDIT HISTORY
C          MSCarey         28-feb-83  AUTHOR
C*FORMAL PARAMETERS
Cin      dum      dummy
C*COMMON BLOCKS
C        none
C*CALLER  various
C*METHOD
C  Calls calendar intrinsic, converts to month-day-year, and packs
C  output variable.
C*LOCAL VARIABLES
C        date     date as returned by intrinsic
C        year     year
C        days     day in year
C        td       a running total of days
C        month    month of year
C        dayom    day of month
C**
```

```
C     RDFSTR**************************************************
      CHARACTER*8 FUNCTION rdfstr(realdt)
      real realdt
C*                                              *** ABSTRACT ***
C#PURPOSE Relate real Date Format to STRing format conversion.
C Converts dates stored in real variables as YYMMDD to a string
C format of "MM/DD/YY".
C#AUDIT HISTORY
C        MSCarey          26-feb-83  AUTHOR
C#FORMAL PARAMETERS
Cin      realdt   date stored in RELATE real variable format
C#COMMON BLOCKS
C        none
C#CALLER  various
C#METHOD
C  Break out the three 2-integer fields and convert them to strings.
C#LOCAL VARIABLES
C        string     character buffer for date
C##
```

```
C     RDLN ****************************************************
$CONTROL SEGMENT=READ
      SUBROUTINE RDLN (IUNIT,LINE,EOF)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE IOC
      INTEGER IUNIT
      LOGICAL EOF
      CHARACTER LINE*LLINE,BUFFER*LLINE
C*                                           *** ABSTRACT ***
C#PURPOSE read a line from IUNIT, without uppercasing
C#AUDIT HISTORY
C         MEMutchler          17 JAN 83  AUTHOR
C         MEMutchler           8  FEB 83 TESTER (program treadl)
C         MSCarey             10  FEB 83 Reads 80 from 5, else 72 col
c         MSCarey              1 MAR 83  Echoes input if cfecho true
C         MSCAREY              5 Mar 83  Handles com file termination
C#TYPE    mnurun utility
C#FORMAL PARAMETERS
Cin       iunit   unit number from which to read
Cout      line    line that was read
Cout      eof     true iff eof was read
C#COMMON BLOCKS
Cin       incpar  global parameter statement
Cin       comcfl  holds command file info.
C#METHOD  An unformated read is done from unit =
C         iunit.  EOF = false unless an end of file is read
C         in which case EOF = true.  If command file building
C         is in use,  LINE is echoed to unit = icomfile.
C         JUST LIKE READLN WITHOUT UPPERC
C#LOCAL VARIABLES   none
C##
```

```
C     RDLNC*****************************************************
$CONTROL SEGMENT=READ
      SUBROUTINE RDLNC (IUNIT,LINE,EOF)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE LPRNTS
       LOGICAL EOF
       INTEGER IUNIT
       CHARACTER LINE*LLINE
C*                                                *** ABSTRACT ***
C#PURPOSE read from file IN and keep track of lines read
C         without uppercasing, especially for reading text files
C#AUDIT HISTORY
C         MEMutchler            17 JAN 83  AUTHOR
C         MEMutchler            8  FEB 83 TESTER  (program treadc)
C#TYPE    mnugen utility
C#FORMAL PARAMETERS
Cin       iunit   file number from which to read
Cout      line    input line read
Cout      eof     true iff eof read from iunit
C#COMMON BLOCKS
Cin       incpar  global parameter statementa
Cin       reads   holds iline
C#METHOD. An unformated read is done from unit =
C         iunit.  EOF = false unless an end of file is read
C         in which case EOF = true.  If command file building
C         is in use,  LINE is echoed to unit = icomfile.
C         Icount is incremented.
C#LOCAL VARIABLES
C         recch   '%' recognition character for comment card
C##
```

```
C     RDLNCU*************************************************
$CONTROL SEGMENT=READ
      SUBROUTINE RDLNCU (IUNIT,LINE,EOF)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE LPRNTS
       LOGICAL EOF
       INTEGER IUNIT
       CHARACTER LINE*LLINE
C*                                            *** ABSTRACT ***
C#PURPOSE read from file IN and keep track of lines read
C#AUDIT HISTORY
C         MEMutchler          17 JAN 83  AUTHOR
C         MEMutchler          8  FEB 83 TESTER  (program treadc)
C#TYPE    mnugen utility
C#FORMAL PARAMETERS
Cin       iunit   file number from which to read
Cout      line    input line read
Cout      eof     true iff eof read from iunit
C#COMMON BLOCKS
Cin       incpar  global parameter statementa
Cin       reads   holds iline
C#METHOD. An unformated read is done from unit =
C         iunit.  EOF = false unless an end of file is read
C         in which case EOF = true.  If command file building
C         is in use,  LINE is echoed to unit = icomfile.
C         Icount is incremented.
C#LOCAL VARIABLES
C         recch   '%' recognition character for comment card
C##
```

```
C     READLN**************************************************
      SUBROUTINE READLN (IUNIT,LINE,EOF)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE IOC
      INTEGER IUNIT
      LOGICAL EOF
      CHARACTER LINE*LLINE,BUFFER*LLINE
C*                                           *** ABSTRACT ***
C#PURPOSE read a line from IUNIT
C#AUDIT HISTORY
C        MEMutchler          17 JAN 83  AUTHOR
C        MEMutchler          8  FEB 83 TESTER (program treadl)
C        MSCarey            10  FEB 83 Reads 80 from 5, else 72 col
c        MSCarey             1 MAR 83  Echoes input if cfecho true
C        MSCAREY             5 Mar 83  Handles com file termination
C#TYPE   mnurun utility
C#FORMAL PARAMETERS
Cin      iunit   unit number from which to read
Cout     line    line that was read
Cout     eof     true iff eof was read
C#COMMON BLOCKS
Cin      incpar  global parameter statement
Cin      comcfl  holds command file info.
C#METHOD An unformated read is done from unit =
C        iunit.  EOF = false unless an end of file is read
C        in which case EOF = true.  If command file building
C        is in use,  LINE is echoed to unit = icomfile.
C#LOCAL VARIABLES   none
C##
```

```
C     RTRIM**************************************************
$CONTROL check=2
      INTEGER FUNCTION rtrim(string,length)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER length
      CHARACTER*1 string(length)
C*                                          *** ABSTRACT ***
C#PURPOSE Finds length of string, NOT including any trailing blanks.
C#AUDIT HISTORY
C         Densmore        28-Oct-82  AUTHOR
C#TYPE    Simple function
C#FORMAL PARAMETERS
Cin       string  the character string
Cin       length  length of string
Cfunction rtrim   length of string without trailing blanks
C#METHOD
C  Uses HP Fortran substring operator to locate last nonblank.
C  If all blank, rtrim is returned zero.
C##
```

```
C     SCLEAR**************************************************
$CONTROL SEGMENT=devctrl
      SUBROUTINE SCLEAR
C*                      *** FORMAL PARAMETER DECLARATIONS ***
C*                                           *** ABSTRACT ***
C#PURPOSE clear terminal screen
C#AUDIT HISTORY
C         MEMutchler      11-mar-83  AUTHOR
C#TYPE    mnurun utility
C#FORMAL PARAMETERS     none
C#COMMON BLOCKS
Cin       envirn  holds clear screen control characters
Cin       ioc     i/o file assignments
C#CALLER  display menu routines
C#METHOD  write control characters to unit iout
C##
```

```
C     SETCCL*************************************************
$CONTROL SEGMENT=devctrl
      SUBROUTINE SETCCL
C*                       *** FORMAL PARAMETER DECLARATIONS ***
C*                                              *** ABSTRACT ***
C#PURPOSE Set clear screen control characters
C#AUDIT HISTORY
C         MEMutchler      11-mar-83  AUTHOR
C#TYPE    murun utility
C#FORMAL PARAMETERS none
C#COMMON BLOCKS
Cin       pvalue  holds runtime parameter values
Cin       pvdecl  holds declarations for parameter names
Cin       pveqiv  equivalence statements between pvdecl and pvalue
Cio       envirn  holds info about runtime enviornment
C#CALLER  sclear
C#METHOD
C  set correct characters according to terminal type
C#LOCAL VARIABLES
C         none
C##
```

```
C     SETTTY**************************************************
$CONTROL SEGMENT=devctrl
      SUBROUTINE SETTTY
C*                      *** FORMAL PARAMETER DECLARATIONS ***
C*                                            *** ABSTRACT ***
C#PURPOSE Determine user's terminal type so that control
C         characters con be set accordingly.
C#AUDIT HISTORY
C         MEMutchler      12-MAR-83  AUTHOR
C#TYPE    mnurun utility
C#FORMAL PARAMETERS    none
C#COMMON BLOCKS
Cin       pvalue  holds runtime parameter values
Cin       pvdecl  holds declarations for parameter names
Cin       pveqiv  equivalence statements between pvdecl and pvalue
C#CALLER  inimnu
C#METHOD
C  determine terminal type by checking terminal line number, if
C  it is 51, the terminal is an hp, if not set it to the most
C  commonly used terminal at DSA
C#LOCAL VARIABLES
C         filnum  file number assigned to logical unit 6
C         linum   terminal line number
C##
```

```
C     SLPRNT************************************************
$CONTROL check=3
      SUBROUTINE slprnt(nprnt,vprnt)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER nptnr
      LOGICAL vprnt
C*                                            *** ABSTRACT ***
C#PURPOSE alters specific LPRNTS values
C#AUDIT HISTORY
C         Densmore        10-Feb-83  AUTHOR
C#TYPE    miscellaneous utility
C#FORMAL PARAMETERS
Cin       nprnt   lprnts index
Cin       vprnt   lprnts value (true or false)
C#COMMON BLOCKS
Cout      lprnts  diagnostic flags
C##
```

```
C     STOPCF ************************************************
$CONTROL SEGMENT=MENU
      SUBROUTINE stopcf
C*                                                   *** ABSTRACT ***
C#PURPOSE Takes care of housekeeping on eof in current command file.
C#AUDIT HISTORY
C        MSCarey          27-FEB-83  AUTHOR
C#FORMAL PARAMETERS
C#COMMON BLOCKS
Cio      comcfl   command file usage status info
Cio      ioc      io unit assignments
C#CALLER readln
C#METHOD
C  Reduces execution nesting level by one. If it reaches zero, sets
C  inuse to false.  Resets io unit numbers.
C#LOCAL VARIABLES
C        none
C##
```

```
C     STRN****************************************
$CONTROL SEGMENT=MENU
      CHARACTER*72 FUNCTION STRN( INUM,LEN)
%INCLUDE INCPAR
      INTEGER LEN, INUM
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER STRING*LLINE
C*                                          *** ABSTRACT ***
C#PURPOSE like intrinsic funcion str, returns len necessary
C#AUDIT HISTORY
C         MEMutchler          1 FEB 83  AUTHOR
```

```
C     TRECOL***********************************************************
$CONTROL check=2
      SUBROUTINE trecol(list,nchar,len,unit)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER nchar,len,unit
      CHARACTER*(nchar) list(len)
C*                                                   *** ABSTRACT ***
C#PURPOSE Prints LIST in three columns on unit UNIT
C#AUDIT HISTORY
C         Densmore        10-Jun-83  AUTHOR
C#TYPE    I/O utility
C#FORMAL PARAMETERS
Cin       list    list of strings to print
Cin       nchar   length of each string
Cin       len     number of strings
Cin       unit    Logical Unit Number on which to print strings
C#METHOD  simple write statement
C##
```

```
C     TTYINI*************************************************
$CONTROL check=3
      SUBROUTINE ttyini
C*                                              *** ABSTRACT ***
C#PURPOSE initializes /tty/
C#AUDIT HISTORY
C         Densmore         24-Mar-83  AUTHOR
C#TYPE    screen utility
C#COMMON BLOCKS
Cin/out   tty     terminal parameters
C##
```

```
C     UPPERC******lowerc****************************************
$CONTROL check=2
      SUBROUTINE upperc(text,n)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER n
      CHARACTER*1 text(n)
C*                                          *** ABSTRACT ***
C#PURPOSE Convert all lower (upper) case characters to upper (lower)
C#AUDIT HISTORY
C         Densmore        27-Oct-82  AUTHOR
C#TYPE    Inout Subroutine
C#FORMAL PARAMETERS
Cin/out   text    text string to be modified
Cin       n       length (characters) of text string
C#METHOD
C  Uses HP's byte addressing construct to move through the
C  string and locate any in the appropriate range.  Since
C  in ASCII the difference between any lowercase letter and
C  the corresponding uppercase letter is a constant value
C  (decimal 32), this value is merely added or subtracted
C  from the integer representation of each character to be
C  altered to opposite case.
C##
```

```
C     UREAD*******uwrite***************************************
$CONTROL check=2
      SUBROUTINE uread(unit,plist,length)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER unit,length,plist(length)
C*                                             *** ABSTRACT ***
C#PURPOSE Allows record structure unformatted reads and writes
C#AUDIT HISTORY
C          Densmore        04-Apr-83  AUTHOR
C#TYPE     I/O Utility
C#FORMAL PARAMETERS
Cin        unit    logical unit number for transfer file
Cin/out    plist   parameter list of *2 words (in=write, out=read)
Cin        length  length of plist
C##
```

```
C     USRINF*************************************************
      SUBROUTINE usrinf(uname,ugroup,uacct,uhome)
      character*8 uname,ugroup,uacct,uhome
C*                                             *** ABSTRACT ***
C#PURPOSE retrieves user name/directory info
C#AUDIT HISTORY
C         Densmore        13-jan-83  AUTHOR
C#FORMAL PARAMETERS
Cout      uname   user name
Cout      ugroup  user's log-on group
Cout      uacct   user's log-on account
Cout      uhome   user's home group, if any
C#COMMON BLOCKS
C         none
C#CALLER  various
C#METHOD
C  See fortran manual appendix A for discussion of calls to intrinsics
C  See MPE Intrinsics manual page 2-195 for "WHO"
C#LOCAL VARIABLES
c         none
C##
```

```
C     VSUMNI******vsubni***************************************
$CONTROL check=3
      SUBROUTINE vsumni(n,head,tail,result)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER n
      INTEGER head(n),tail(n),result(n)
C*                                           *** ABSTRACT ***
C#PURPOSE vector sum/difference for integers
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       n       length of vectors
Cin       head    first vector
Cin       tail    second vector
Cout      result  head+tail  or  head-tail
C##
```

```
C     XMIT/B*************************************************
$CONTROL check=2
      SUBROUTINE xmit (length,source,target)
      INTEGER length
      INTEGER*4 source(1),target(1)
C#PURPOSE fills target array with source via copy
C          "B" entry allows right-shifts by "DO"ing backwards.
C#AUDIT HISTORY   Densmore   28-Oct-82
C#FORMAL PARAMETERS
Cin        length  size of move...if <0, only source(1) is used
Cin        source  source array
Cout       target  target array
C##
```

```
C     XMIT2/B***********************************************
$CONTROL check=2
      SUBROUTINE xmit2(length,source,target)
      INTEGER length
      INTEGER*2 source(1),target(1)
C#PURPOSE fills target array with source via copy
C         "B" entry allows right-shifts by "DO"ing backwards.
C#AUDIT HISTORY   Densmore   28-Oct-82
C#FORMAL PARAMETERS
Cin       length  size of move...if <0, only source(1) is used
Cin       source  source array
Cout      target  target array
C##
```

```
C     XMIT4/B**************************************************
$CONTROL check=2
      SUBROUTINE xmit4(length,source,target)
      INTEGER*4 length
      INTEGER*4 source(1),target(1)
C#PURPOSE fills target array with source via copy
C         "B" entry allows right-shifts by "DO"ing backwards.
C#AUDIT HISTORY   Densmore   28-Oct-82
C#FORMAL PARAMETERS
Cin       length  size of move...if <0, only source(1) is used
Cin       source  source array
Cout      target  target array
C##
```

```
C     XMITC *****xmitb****************************************
$CONTROL check=2
      SUBROUTINE xmitc(length,source,target)
      INTEGER length
      CHARACTER source(1),target(1)
C#PURPOSE fills target array with source via copy
C         "B" entry allows right-shifts by "DO"ing backwards.
C#AUDIT HISTORY   Densmore   28-Oct-82
C#FORMAL PARAMETERS
Cin       length  size of move...if <0, only source(1) is used
Cin       source  source array
Cout      target  target array
C##
```

```
C     YESNO**************************************************
      logical function yesno(in,iout)
      integer in,iout
C*                                            *** ABSTRACT ***
C#PURPOSE Prompts for an answer, true if yes.
C#AUDIT HISTORY
C        MSCarey          03-feb-83  AUTHOR
C#TYPE    I/O utility
C#FORMAL PARAMETERS
C         in        unit number to read from
C         in        unit number to write to
C#COMMON BLOCKS
C#CALLER  various
C#METHOD
C  Prompts, accepting only "y","n","yes", or "no"
C#LOCAL VARIABLES
C        answr    input buffer
C        answer   user answer
C        len      string size
C##
```

```
C      ZABORT***********************************************
      SUBROUTINE zabort
C*                                                *** ABSTRACT ***
C#PURPOSE Aborts when an error occurs...often called by MABORT.
C#AUDIT HISTORY
C         Densmore        27-Oct-82  AUTHOR
C#TYPE    Simple subroutine
C##
```

10.2 FORTRAN UTILITIES FOR DBMS USAGE

The routines in the Data Base management system InterFace library (DBIF; source in dbifa.src, dbifdm.src, dbifl.src, dbifrv.src; principal include files strngs.incl, cursrs.incl; object code in dbif.obj) were created for two reasons:

1) A move of ALIAS software to a computer other than the HP 3000 was declared to be a possibility at the outset of ALIAS development. Given that RELATE runs only on the HP 3000, and that ALIAS routines would be making very heavy use of RELATE, it seemed prudent to buffer all requests for DBMS services through a set of interface routines. At conversion it should be possible to change only the internals of the interface to work with a new DBMS, making it possible to avoid major changes to the applications programs.

2) The RELATE Host Language Interface routines are rather difficult and finicky to work with directly. A more programmer-friendly means of accessing the data base was desired.

10.2.1 DBIF Organization

Although the DBIF can be used to issue any RELATE, CREATE, or GRAF command, it is primarily designed to make use of the routines of RELATE's Host Language Interface. These routines provide a record-level method of data base access (as opposed to the set-level method of interactive RELATE); that is, operations are performed on data base files one tuple at a time. In addition to the obvious read, add, delete, and update capabilities, the HLI also provides a "point" routine which allows the programmer to jump to the location on an index whose field values match the values of the target he specifies. Also, a query routine will return information about relations and the state of the DBMS, and an error routine can be used to learn more about errors after they have occured.

The DBIF can be divided into high-level and low-level routines, the high-level routines being those called by programmers. An annotated listing of the high-level routines is presented in Table 10-5. Low level routines are listed in Table

Table 10-5. DBIFA High-Level Routines

ROUTINE	PURPOSE
CKWPRV	Security utility which module authors can call during their initialization code to see if the user is going to have write access to all the relations he will require. If not, graceful termination can be engineered. This routine duplicates the logic of rvscen, which is called by DBIF routines about to do a DB write to check privelege. Rvscen invokes a ZABORT if priveleges are insufficient, which is why explicit testing is nice.
RCINIT	Initializes DBIF. Must be called before any other DBIF routine.
RELCOM	Call this to execute any RELATE, CREATE, or GRAF command programmatically. Allows simulation of interactive use of RELATE.
RTPADD	Adds a single record to the path specified.
RTPCAL	Requires a buffer of target values for the fields on the current index. Performs a "point" operation which locates the record with fields matching these values; then returns the contents of the record into a second data buffer. Much more efficient than SELECT for many types of searching. Returns no data if point fails.
RTPDEL	Deletes the current record on the path specified.
RTPKIL	Similar to rtpcal, except for record deletion. Finds the record matching the specified index value via a point, then deletes it.
RTPNEW	Attempts to add a record to the current path; this routine is an integer function which returns a status code value, where 0=success, 1=failure due to unary key violation, and 2=failure due to relation full. It's a good idea to use this rather than rtpadd and to place error handling logic in your code.
RTPNFD	Like rtpcal except expects NOT to find the record pointed to. Reads and returns whatever record the point left the record pointer at.
RTPNXT	Reads and returns the next record on the current path.
RTPREP	Like rtpcal except for update. Points to the record specified, then updates it with the specified values.

Table 10-5. DBIFA High-Level Routines

ROUTINE	PURPOSE
RTPUPD	Updates the current record on the specified path with given values.
RVCLOS	Closes a path and de-allocates its cursor.
RVCFIL	Creates a new relation with the specified structure, returning the index of its cursor/path.
RVCKIL	Deletes the relation open on the specified path.
RVCPTH	Creates a relation and opens it with an alternative path name. Very similar to rvcfil.
RVCREL	Opens a relation, returning the index of its cursor/path.
RVCRWD	Rewinds the specified path.
RVCSLC	Does a SELECT and returns the index of its cursor/path. The files the select draws on must already be open on other cursors. Their names must be included as part of normal SELECT syntax somewhere in the field list or WHERE clause (e.g. field list of "+filea.fld1, filea.fld2,fileb.fld9+").
RVCSRT	Opens a relation and sets to a specified index. Note that if points will be desired using only a subset of the index fields (e.g. only SCENARIO of an index on SCENARIO,CLASS) then the last desired field should be followed by a "|" rather than a "," (e.g. SCENARIO|CLASS).
RVCSTS	Opens a relation and sets to a specified index, using an alternative path name. Like rvcsrt.
RVCSYN	Opens a relation on an alternative path name. Similar to rvcrel. Note that the alternative path name routines are seldom useful. Since all files are opened on separate cursors, and the same file may be open more than once under the same name if the opens are done on separate cursors, then there is no particular reason to use path synonyms.

10-6. Section 10.2.7 (subroutine abstracts) gives the detailed calling requirements for each routine.

10.2.2 Using the Routines

The reader will notice that the routines named in Table 10-5 can be divided into four categories by their names:

1) rcinit

2) relcom

3) All routines whose names begin with "rvc".

4) All routines whose names begin with "rtp".

Rcinit initializes the DBIF. Relcom can be used to give any interactive RELATE, CREATE, or GRAF command programmatically. The "rvc" (Relate Virtual Cursor) routines are used to set up a retrieval path; they open and close files, choose indexes, and give selects. The "rtp" (Relate TuPle) routines find and/or manipulate individual records in data relations.

A typical calling sequence would include rcinit, rvcsrt to open a relation and set to a particular index, and a combination of rtpcal and rtpnxt calls to jump to a location on the index and retrieve records starting there.

There are four basic choices for setting up a retrieval path:

1) The equivalent of a regular OPEN FILE (rvcrel).

2) The equivalent of an OPEN FILE followed by a SET INDEX (rvcsrt).

3) The equivalent of a SELECT (rvcslc); this presumes that the relations the SELECT wants have already been opened by other "rvc" calls.

4) The equivalent of a CREATE FILE (rvcfil).

Table 10-6. DBIFA Internal (Low-Level) Routines

ROUTINE	PURPOSE
DCGIDX	Does a SET INDEX and related data structure set-up.
DCGTUP	Does a record update for the current tuple.
DCINIT	Initializes the cursor and string chain subsystems.
DCKCRS	Error management routine which prints status information which can be extracted from the cursor the problem occurred on and from the DBIF data structure.
DCKERR	Checks to see if an error happened on the last call involving the given cursor, and causes the associated RELATE error message to be printed to the terminal if one did.
DCSLCT	CLOSES all files open on a cursor a select was given on and releases the cursor.
DCSPTH	CLOSES the file open on a regular (non-select) cursor and releases the cursor.
DELCRS	Routine which actually releases cursors, both in DBIF data structure and in RELATE son process.
DELIDX	Deletes an index from the current relation. This routine is non-function (i.e. it will abort) as long as the current convention of opening all relations with MODE=SHARED is in place.
DELREL	Deletes the relation open on the given cursor.
DELTUP	Deletes the current record in the relation open on the given cursor.
DMKCRS	Allocates a cursor in the DBIF (/cursrs/) data structure and call rdbinit to initialize it.
DMKIDX	Attempts to create a new index. Will always abort at present since all relations are opened with MODE=SHARED.
DMKREL	Creates a new relation.
DMKTUP	Adds a record to the specified path.
DOPPTH	Opens the given relation on the given cursor.
DPCMD	Diagnostic print utilities.

Table 10-6. DBIFA Internal (Low-Level) Routines

ROUTINE	PURPOSE
DPCMD1	
DREWND	Rewinds the file open on the given cursor.
DSOPEN	Service routine for rvcslc, opens all the files requested as part of the select on the select's cursor. Note that since only the path parts of the file names are specified in the select syntax the group names must be extracted from the DBIF data structure and by rdbinfo calls.
DSELCT	Does a select.
DTCALC	Does a point on the given cursor.
DTNEXT	Reads a record from the given cursor's path.
LENIDX	Figures out the number of words in the specified index and stores it.
RCKPRV	Checks to see if the user has write priveleges on the given file. Both sysusr.sysro and scenario system checks implicitly involved. Called before each DBIF write operation as a last-ditch defense.
RSTIDX	Execute for index-setting.
RVSCEN	Security check and flag-setting routine called whenver a path is set up by one of the "rvc" routines. Also sets scenario key field value in /scenar/.
SNRLSN SNRLNM	Utilities used to access the scenario system's extra data segment. Truly part of the scenario system, as is rvscen, but present here as part of scenario system "presence" in DBIF.

The remaining utilities perform similar actions but using alternative path names (rvcsyn, rvcsts, and rvcpth respectively), close files, delete files, and "rewind" record pointers to top-of-file.

Use of the relcom routine should be avoided except to give commands not provided for in the other utilities.

Developers are likely to find the rtpcal routine particularly useful. Functionally similar to a combination of a BUILDER RECORD POINT and RECORD READ given in sequence, this routine can locate and return the contents of a particular record in a relation (by key/index value) much faster than an equivalent select can. Benchmarks have shown that rtpcal requires approximately 250 milliseconds (single-user) regardless of the size of the relation or the number of fields in the index.

Note that if it is necessary for the implicit point to operate on only a subset of an index (e.g. you want to point only to YARD on an index of YARD,DATADATE,ENTRY_DATE) this can be done by specifying the index with a "|" rather than a "," following the last field of point-interest in the "rvc" call (e.g. YARD|DATADATE,ENTRY_DATE)

All of the routines require a single-word integer argument called "cursor". More about this in the next section.

In addition to a cursor index, the "rvc" routines often require one or more delimited text strings which specify the name of the relation to be opened, fields in the index to be set to, clauses to include in the select statement, etc. The only unusual requirement is by rvcslc, which requires that the name of each relation to selection is to draw from be mentioned at least once in the field list or by clause argument, in the form "relation.fieldname,relation.fieldname,....".

The "rtp" routines will typically require at least one data buffer as an argument, and perhaps a delimited list of fields to be returned, updated, etc. It is VERY IMPORTANT that the DATA BUFFER BE WORD ALIGNED, i.e. that it not be a character string. If the data is of type character, equivalence the character variable to an integer array and pass the integer form as an argument. RELATE will abort nastily if it received a non-word-aligned buffer.

To be most usable, data buffers should consist of a series of variables, one per corresponding field in the relation (or on the index), and of identical length and type in comparison to the fields. Thus a buffer for the fields SCENARIO,CLASS,HULL would consist of a character*12,character*10,integer*2 series of variables, all next to each other in process data memory. The best way to ensure that the variables are actually sequential in memory is to declare them sequentially in the same common block (character and numeric data may be mixed in HP FORTRAN common blocks). Equivalences may also be used, but require more coding. Note in the example given that the "scenario" variable needs to be equivalenced to an integer array to word-align the common block.

When using the rtpcal routine, it may seem that there should be two field lists as arguments, to accompany the two data buffers required: one to specify the fields in the target to be pointed to, and one to specify the fields to be returned. The target field list is implicit, being defined by the current index. Note that rtpcal will ALWAYS return notfnd=.true. if the target (key) data buffer is not of the same length as the index, or if values are improperly positioned within the buffer. Note especially that since RELATE left-justifies strings, they should be left-justified in the target buffer (but non-justified strings can be placed in the relation using the DBIF).

10.2.3 Cursors and the DBIF Data Structure

The DBIF manipulates three global data structures. The first is a string buffer, managed by the CHN___ general purpose utilities, which is used for handling field lists, file names, etc. This buffer is of no particular interest to users; the chain strategy was used since field lists can exceed 255 characters and in order to conserve memory.

The second data structure is the cursen array in the /scenar/ common block. When a file is opened using any of the "rvc" routines, the proper scenario key field value for that file is retrieved and placed in the location in cursen indexed by the cursor index to be returned to the "rvc" routine's caller. The corresponding location in the wrtprv array is also set. This activity actually is the portion of the scenario system which resides in the DBIF.

The "cursor" data structure is the third. Remember from Section 8.4 that HLI routines require that a 50-word integer array be provided with each call as a communication area and a repository for certain data the HLI needs to have global. These arrays are called "partitions" in BUILDER; they are called "cursors" in the HLI section of the RELATE manual.

The DBIF has the capacity to work with 20 cursors. The DBIF is designed so that each retrieval path will have its own cursor; except for paths set up by an rvcslc (select) call, a SHOW PATH command given on any of these cursors would reveal only a single file open. Since paths set up by rvcslc may only use files open on another cursor, this means that the DBIF may work with no more than 20 files simultaneously.

The 50-word integer arrays are managed internally to the DBIF. The "rvc" routines are all integer functions which return a single word integer with a value between 1 and 20---a cursor index. Calls to other DBIF routines supply this index to indicate which file they want to work with; the index is then used to pick out a particular element of the DBIF's 20x50 cursor array.

Thus, the information returned by the "rvc" routines in response to a path-creation call is useable only in queries and updates made through the "rtp" (and relcom) routines.

This design makes it unnecessary for application routines to create and manage large cursor data structures, and also makes intensive work with a few relations easier since the file name and index need be specified only once; after that only an integer variable is required in calling code.

This intensive use of a few relations is the most common form of programmatic data base access.

The design is limiting in that only 20 files can be open simultaneously, but remember that a single RELATE process can handle a maximum of about 25 open files before aborting with a memory overflow. The rdbinitx means of using multiple RELATE sons was not available at the time the DBIF was implemented.

Giving the DBIF a multiple-son handling capability would require paging of the cursor data structure as well as substitution of rdbinitx calls for rdbinit calls (and logic to detect when to use a new process as opposed to an old one). Otherwise the cursor data structure would begin to take up too much process memory in the Core.

10.2.4 <u>DBIF Internals</u>

Many high-level DBIF routines just call low-level routines which in turn call functionall similar HLI routines. For example, rtpupd calls dcgtup which calls rdbupdate. Given an understanding of the HLI, the structure of the DBIF is thus fairly clear. However, string handling, error handling, and index management require some exposition.

10.2.4.1 String Handling

As noted above, the DBIF uses the string chain (CHN___) general purpose utilities to manage a string buffer. This buffer, called str, is 3K bytes long, and stored in the /strngs/ block. A typical DBIF routine will receive a field list in the form a delimited string in a character variable. The field list must be left-justified in a word-aligned array for passage to RELATE, and must be uppercased. The routine will move the list into a (word-aligned) area of str via a call to the lstrng un-delimit utility, will uppercase the entire area, and will then pass name of the integer array equivalenced to str to the given HLI routine.

10.2.4.2 Error Handling

After every call to an HLI routine the DBIF uses dckerr to check to see if an error occurred during HLI execution. If one did (indicated by a non-zero value of the first word of the appropriate cursor), then the HLI routine rdberror is called with a request to print the RELATE error message corresponding to the problem which occured, and dckcrs is called to print the status of some DBIF variables.

This error handling is one of the greatest benefits of using the DBIF, since any errors which occur are guaranteed detection and an at least moderately explicable error message.

Note that the DBIF uses lprnts 2 and 3, and that quite extensive running diagnostics of DBIF operations are generated if these are both set to .true.

10.2.4.3 Index Management

When a user specifies an index in an rvcsrt, rvcsts, or rvcslc call several things must happen, all of which are managed by the rstidx routine. First, an attempt is made to do a SET INDEX via a call to dcgidx. The method used is to query RELATE for the indexes on the open file, doing the SET for the first one which has at least fields matching the keys requested. Note that if the request is for SCENARIO,CLASS, and the two indexes on the file are SCENARIO,CLASS,HULL and SCENARIO,CLASS, the first index will be the one chosen.

If this fails, the routine will attempt to create an index. This creation will always fail, since all relations accessed through the DBIF are opened with MODE=SHARED, and indexes can only be created when the user has exclusive access. Thus, a permanent index must exist which matches the request.

After a successful set, the length of the index fields in words is determined via a call to lenidx. This will be needed if rtpcal is ever called on the given relation, because the number of words in the index to use is an argument to the rdbpoint routine.

There are two cases in which this length will not just be the length of the actual index used. The first case occurs when, as in the example above, the number of words in the requested index is less than the number in the index used, because there are "superfluous" fields in the actual index. It is very important that the argument to rdbpoint have the number of words implicitly requested in this case: since the programmer has no idea which index will be chosen by the DBIF, he will have contructed his target buffer for rtpcal to be of length matching only those fields in his index request. If the actual index length were used, his points would always fail.

The second case occurs when the programmer wants to point on only a subset of the index requested. He can do this by replacing the comma following the last field he want included in the point with a "|" in the index request he makes to rvcsrt, rvcsts, or rvcslc. The number of words stored for use in rdbpoint calls must in this case match the size of the fields named before the "|". For example, a request for index SCENARIO,CLASS|HULL would yield a word count of 11, not 12.

10.2.5 Security and the Scenario System "Presence"

In addition to the mechanics of data base access, the DBIF is also concerned with security enforcement. In particular, it is the last line of defense against unauthorized programmatic data base changes (before RELATE security). Every DBIF routine which modifies the contents of relations calls the rckprv utility to check the user's priveleges before doing so.

Changes may be disallowed (leading to a ZABORT) for two reasons: the user does not have basic DB change priveleges, as specified by the ALTDB flag in the sysusr.sysro relation; or the scenario the user is currently working with is using the given relation's data indirectly, an access method which forbids changes.

Developers should call the ckwprv logical function when they open a relation to see if the user will have write priveleges, and abort gracefully if not.

The scenario system is also supported by calls to the rvscen routine by all the "rvc" routines when they open a relation. Rvscen retrieves the proper scenario field key value for the current scenario for the given relation from the scenario system's extra data segment and places it in the appropriate (cursor_index) location in the cursen array of the /scenar/ common block for referening by application routines.

10.2.6 DBIF Modification

If it should be necessary to modify an recompile any DBIF routines, be sure to re-create the dbif.obj file when compilation is complete. Typically, simple compilation of any portion of the

DBIF will not result in changes to dbif.obj (e.g. compiling dbifa.src with the normal utilities will create or update dbifa.obj). Two re-create dbif.obj from the four constituent DBIF source libraries, use the command "GLUE dbif" at the MPE level.

Note that any new routines should always be assigned to segment dbif.

10.2.7 DBIF Subroutine Abstracts

```
C     CKWPRV ***********************************************
$CONTROL check=3,segment=dbif
      LOGICAL FUNCTION ckwprv(modnam,filnam)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      character*20 modnam,filnam
C*                                              *** ABSTRACT ***
C#PURPOSE Checks to see if user has write privelege for the
C  file named for the current scenario.  Useful at top of
C  module initialization.  Duplicates logic of rvchek.
C#AUDIT HISTORY
C          MSCarey         20-sep-83  AUTHOR
C#FORMAL PARAMETERS
Cin       modnam  delimeted name of module test being performed
C                 for.  If of nonzero length, ckwprv writes warning on
C                 lack of write privelege.
Cin       filnam   name of DB file to test priveleges for
C#COMMON BLOCKS
Cin       uzrprv  user privelege levels
Cin       scenar  scenario status info
Ci n       snrref  scenario set-up info and function declarations
C#CALLER  various
C#METHOD
C  Uses logic similar to rvscen.  Find name of file in list of
C  known DB files, then check to see if scenario field value for
C  that file matches current scenario overall name.  Also check
C  overall user privelege levels.
C##
```

```
C     DCGDMN*************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE dcgdmn(relatn,flist,fmtlst)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*255 relatn,flist,fmtlst
C*                                        *** ABSTRACT ***
C#PURPOSE change domain of a relation
C#AUDIT HISTORY
C         Densmore        12-Dec-82  AUTHOR
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
Cin       relatn  DTS relation name
Cin       flist   DTS field list
Cin       fmtlst  DTS format list
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#METHOD
C  not currently allowed
C##
```

```
C     DCGIDX**************************************************
$CONTROL segment=dbif,check=3
      LOGICAL FUNCTION dcgidx(cursor,flist)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*255 flist
      INTEGER cursor
C*                                           *** ABSTRACT ***
C#PURPOSE sets or ChanGes InDeXes to an already open file
C#AUDIT HISTORY
C         Densmore        12-Dec-82  AUTHOR
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
Cin       cursor  cursor index to an open path
Cin       flist   DTS field list defining desired index
Cfunction dcgidx  .TRUE. if the desired index is found
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   strngs  string buffers
C#METHOD
C         Carey           07-Dec-83  Now uses process id-specific
C                                    rdbinitx RELATE init routine
C  performs SET INDEX <fieldlist>
C##
```

```
C      DCGTUP*****************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE dcgtup(cursor,flist,source)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,source(1)
      CHARACTER*255 flist
C*                                          *** ABSTRACT ***
C#PURPOSE ChanGes TUPle (modifies)
C#AUDIT HISTORY
C         Densmore       12-Dec-82  AUTHOR
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
Cin       cursor  cursor index
Cin       flist   DTS names of fields to be updated
Cin       source  new values for each of these fields
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   strngs  string buffers
C#METHOD
C  calls rdbupdate
C##
```

```
C     DCINIT*************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE dcinit
C*                                                 *** ABSTRACT ***
C#PURPOSE Database Cursor INITialization
C#AUDIT HISTORY
C         Densmore        15-Dec-82  AUTHOR
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
C         none
C#COMMON BLOCKS
Cout      cursrs  cursor buffers
Cout      strngs  string buffers
C#METHOD
C  initializes all chained buffer systems
C##
```

```
C     DCKCRS***************************************************
      SUBROUTINE dckcrs(cursor,out)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,out
C*                                               *** ABSTRACT ***
C#PURPOSE CheckK CuRSor...prints locally kept cursor information
C#AUDIT HISTORY
C        Densmore        26-Dec-82  AUTHOR
C#TYPE   Database low-level interface utility
C#FORMAL PARAMETERS
Cin      cursor  index to cursor
Cin      out     output logical unit number
C#COMMON BLOCKS
Cin      cursrs  cursor buffers
Cin      indexs  index buffers
C##
```

```
C      DCKERR************************************************
$CONTROL segment=dbif,check=3
      LOGICAL FUNCTION dckerr(cursor)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor
C*                                          *** ABSTRACT ***
C#PURPOSE checks for any errors in relate processing
C#AUDIT HISTORY
C          Densmore         12-Dec-82  AUTHOR
C#TYPE     Database low-level interface utility
C#FORMAL PARAMETERS
Cin        cursor  cursor index
Cfunction dckerr  returns true if an error exists
C#COMMON BLOCKS
Cin        cursrs  cursor buffers
C#METHOD
C  uses rdberror; prints error information on RDBOUT ::= $STDLIST
C  no error exists if first word in cursor buffer is zero
C##
```

```
C     DCSLCT************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE dcslct(cursor)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor
C*                                         *** ABSTRACT ***
C#PURPOSE Database...CloSes current SeLeCT virtual cursor
C#AUDIT HISTORY
C         Densmore         15-Dec-82  AUTHOR
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
Cin/out   cursor  cursor index of cursor associated with the
C                 cursor to be closed
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#METHOD
C  see SELECT command.
C##
```

```
C     DCSPTH*******************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE dcspth(cursor,pthnam)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor
      CHARACTER*255 pthnam
C*                                          *** ABSTRACT ***
C#PURPOSE closes a previously open path and its cursor
C#AUDIT HISTORY
C         Densmore        15-Dec-82  AUTHOR
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
Cin/out   cursor  cursor pointer index
Cin       pthnam  DTS pathname to be closed
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   strngs  string buffers
C#METHOD
C  calls CLOSE PATH relate command
C##
```

```
C     DELCRS***********************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE delcrs(cursor)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor
C*                                          *** ABSTRACT ***
C#PURPOSE closes and DELetes CuRSor
C#AUDIT HISTORY
C         Densmore        12-Dec-82  AUTHOR
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
Cin/out   cursor  cursor index...set to zero indicating deallocation
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#METHOD
C  closes cursor; deallocates cursor index
C##
```

```
C     DELIDX*************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE delidx(cursor,relatn,flist)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*255 relatn,flist
      INTEGER cursor
C*                                              *** ABSTRACT ***
C#PURPOSE deletes an index from an already open relation
C#AUDIT HISTORY
C          Densmore        12-Dec-82  AUTHOR
C#TYPE     Database low-level interface utility
C#FORMAL PARAMETERS
Cin        cursor  index to cursor opened under the pathname RELATN
Cin        relatn  DTS relation name
Cin        flist   DTS field-list defining index
C#COMMON BLOCKS
Cin/out    cursrs  cursor buffers
Cin/out    strngs  string buffers
C#METHOD
C  call set-index using flist; then calls
C  rdbinfo to get index number so it can be deleted.
C##
```

```
C     DELREL*************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE delrel(cursor,relatn)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*20 relatn
      INTEGER cursor
C*                                            *** ABSTRACT ***
C#PURPOSE deletes a relation; assumes that the relation is open
C#AUDIT HISTORY
C         Densmore        12-Dec-82  AUTHOR
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
Cin       cursor  index to cursor opened under the pathname RELATN
Cin       relatn  DTS relation name to delete
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#METHOD
C  Calls for a PURGE FILE command...also deallocates cursor resources
C##
```

```
C     DELTUP*********************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE deltup(cursor)
C*                          *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor
C*                                                    *** ABSTRACT ***
C#PURPOSE deletes the current tuple
C#AUDIT HISTORY
C         Densmore         12-Dec-82  AUTHOR
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
Cin       cursor  cursor index
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#METHOD
C  calls rdbdelete
C##
```

```
C     DMKCRS*************************************************
$CONTROL segment=dbif,check=3
      INTEGER FUNCTION dmkcrs(dummy)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER dummy
C*                                           *** ABSTRACT ***
C#PURPOSE retrieves a unique cursor index from chain data type
C         and initializes the cursor...Database MaKe CuRSor
C#AUDIT HISTORY
C         Densmore        12-Dec-82  AUTHOR
C#TYPE    database low-level interface utility
C#FORMAL PARAMETERS
Cin       dummy   dummy variable
Cfunction dmkcrs  a unique index taken from the crschn chain data
C                 type which indexes a cursor in the array crs of
C                 RELATE cursors.  This integer is used throughout
C                 the RELATE utilities to represent a cursor.
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#METHOD
C  calls chnalo...assumes that chain is initialized; then
C  initializes the indexed cursor via RDBINIT
C##
```

```
C     DMKIDX*************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE dmkidx(cursor,relatn,flist,unary)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor
      CHARACTER*255 relatn,flist
      LOGICAL unary
C*                                          *** ABSTRACT ***
C#PURPOSE Database create (MaKe) InDeX for relate system;
C         the relation named by relatn must be open.
C#AUDIT HISTORY
C         Densmore        12-Dec-82  AUTHOR
C#TYPE    Database Low-level interface utility
C#FORMAL PARAMETERS
Cin       cursor  index to cursor opened under the pathname RELATN
Cin       relatn  DTS name of relation
Cin       flist   DTS field list for indexing purposes
Cin       unary   logical; .TRUE. if no key may be duplicated or
C                 allowed in index
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   strngs  string buffers
C#METHOD
C  forms command string, calls relate.
C##
```

```
C     DMKREL*************************************************
$CONTROL segment=dbif,check=2
      INTEGER FUNCTION dmkrel(relatn,pthnam,struct)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*255 relatn,pthnam
      CHARACTER*1 struct(600)
C*                                          *** ABSTRACT ***
C#PURPOSE Database, MaKes RELation; returns cursor
C#AUDIT HISTORY
C         Densmore        22-Feb-83  Deleted USEPTH arg so that high
C                                    level routines easily interface
C         Carey           10-feb-83  Made struct an array to
C                                    accommodate big field lists.
C         Densmore        12-Dec-82  AUTHOR
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
Cin       relatn  DTS relation name
Cin       pthnam  DTS path name, if not same as relation name
Cin       struct  DTS field name list specifying structure of relation
Cfunction dmkrel  cursor index to the new cursor constructed.
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   strngs  string buffers
C#METHOD
C  creates new cursor; creates command; calls relate.
C##
```

```
C     DMKTUP***********************************************************
$CONTROL segment=dbif,check=3
      INTEGER FUNCTION dmktup(cursor,list,source)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,source(1)
      CHARACTER*255 list
C*                                              *** ABSTRACT ***
C#PURPOSE Database MaKe TUPle -- adds tuple to relation pointed
C         to by cursor.
C#AUDIT HISTORY
C         Densmore        14-Dec-82  AUTHOR
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
Cin       cursor  index to a cursor from the pool
Cin       list    DTS list of fields in tuple to be added
Cin       source  array of data referenced by list to be added as
C                 the new tuple
Cfunction dmktup  returns success index...0 means successful;
C                 1 means unary violation; 2 means file full (EOF)
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   strngs  string buffer variables
C#METHOD
C  word aligns list, then calls rdbadd
C##
```

```
C     DOPPTH************************************************
$CONTROL segment=dbif,check=3
      INTEGER FUNCTION doppth(relatn,pthnam)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*255 relatn,pthnam
C*                                          *** ABSTRACT ***
C#PURPOSE opens a path to the named relation
C#AUDIT HISTORY
C         Densmore        15-Dec-82  AUTHOR
C         Carey            5-may-83  open all files in shared mode
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
Cin       relatn  DTS relation name
Cin       pthnam  DTS path name
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   strngs  string buffers
C#METHOD
C  calls OPEN PATH relate command
C##
```

```
C     DPCMD*****************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE dpcmd(cursor,routin,kstr,len)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*6 routin
      INTEGER len,cursor
C *** INTEGER kstr( (len+1)/2 )
      INTEGER kstr(len)
C*                                         *** ABSTRACT ***
C#PURPOSE For use when LPRNT 3 is on to print RELATE commands issued
C#AUDIT HISTORY
C         Densmore        14-Feb-83  AUTHOR
C#TYPE    diagnostic
C#FORMAL PARAMETERS
Cin       cursor  cursor index
Cin       routin  character*6 routine name
Cin       kstr    integer array containing characters of cmd
Cin       len     number of characters in kstr
C#COMMON BLOCKS
Cin       lprnts  diagnostic flags and ioutp
C#CALLER  all D... relate routines
C##
```

```
C     DPCMD1**************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE dpcmd1(cursor,routin)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*6 routin
      INTEGER cursor
C*                                              *** ABSTRACT ***
C#PURPOSE Like DPCMD, except for zero length character strings
C#AUDIT HISTORY
C          Densmore        14-Feb-83  AUTHOR
C#TYPE     diagnostic
C#FORMAL PARAMETERS
Cin        cursor  cursor index
Cin        routin  routine name
C#COMMON BLOCKS
Cin        lprnts  diagnostic flags and ioutp
C#CALLER   all D... routines
C##
```

```
C     DREWND*************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE drewnd(cursor)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor
C*                                          *** ABSTRACT ***
C#PURPOSE rewinds relation corresponding to cursor
C#AUDIT HISTORY
C          Densmore        02-Feb-83  AUTHOR
C#TYPE    low-level relate database utility
C#FORMAL PARAMETERS
Cin       cursor  cursor to be rewound
C#CALLER  rvcrwd
C#METHOD
C  Calls RDBPOINT with rewind flag set.
C##
```

```
C     DSOPEN************************************************
$CONTROL segment=dbif,check=3
      INTEGER FUNCTION dsopen(crss,ncrss)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER ncrss,crss(ncrss)
C*                                            *** ABSTRACT ***
C#PURPOSE Inits a new cursor and opens files associated with each crs
C#AUDIT HISTORY
C         Densmore        23-Mar-83  AUTHOR
C#TYPE    low level RELATE database utility
C#FORMAL PARAMETERS
Cin       crss    cursor index for each cursor associate with a file
Cin       ncrss   length of crss
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#CALLER  rvcslc
C#METHOD
C  loops over cursors getting current db number; retrieves full
C  filename using rdbinfo again; opens each file on the new cursor
C#LOCAL VARIABLES
C         icrs     do index
C         cursor   each crss value
C         ndb      database number
C         len      length of dbname
C         lcmd     length of command
C         info     info array for rdbinfo (dbname)
C         dbname   full file name for database (info)
C         comand   full RELATE command (icmd)
C         icmd     integer version of (comand)
C##
```

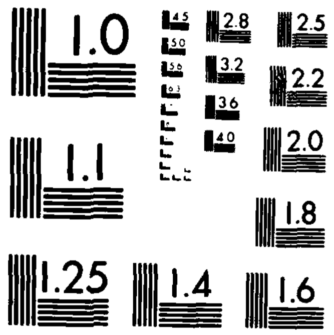
```
C     DSLECT*************************************************
$CONTROL segment=dbif,check=2
      SUBROUTINE dslect(cursor,tgtlst,unique,keylst,cond)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      LOGICAL unique
      INTEGER cursor
      CHARACTER*255 tgtlst,keylst,cond
C*                                          *** ABSTRACT ***
C#PURPOSE implements RELATE's select command
C#AUDIT HISTORY
C         Densmore        15-Dec-82  AUTHOR
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
Cin       cursor  cursor index for cursor on which select is to be done
Cin       tgtlst  DTS target list, indicating what fields should be
C                 returned and the values they should assume; in
C                 the form    name1[=expr][,name2[=expr]]...
Cin       unique  LOGICAL indicates that selection results unique
C                 values in the key list
Cin       keylst  DTS names of fields on which selection is sorted
C                 optional unless unique is TRUE; avoid specification
C                 via the DTS '::'
Cin       cond    DTS condition which created virtual tuples should
C                 be returned.
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   strngs  string buffers
C#METHOD
C  performs RELATE select command
C##
```

```
C     DTCALC**************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE dtcalc(cursor,keyval,notfnd)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,keyval(1)
      LOGICAL notfnd
C*                                            *** ABSTRACT ***
C#PURPOSE Calculates position of next tuple
C#AUDIT HISTORY
C          Densmore        21-Feb-83  Add RDBINFO to retrieve |key|
C          Densmore        18-Feb-83  Remove FLIST,TUPLE arguments
C          Densmore        15-Dec-82  AUTHOR
C#TYPE     Database low-level interface utility
C#FORMAL PARAMETERS
Cin        cursor  cursor index
Cin        keyval  key value to search for
Cout       notfnd  not-found flag -- .TRUE. if tuple not found
C#COMMON BLOCKS
Cin/out    cursrs  cursor buffers
Cin/out    strngs  string buffers
C#METHOD
C  retrieves length of KEYVAL from crsxl, then calls RDBPOINT
C##
```

```
C     DTNEXT**************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE dtnext(cursor,flist,tuple,eof)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,tuple(1)
      CHARACTER*255 flist
      LOGICAL eof
C*                                           *** ABSTRACT ***
C#PURPOSE Returns next tuple associated with cursor
C#AUDIT HISTORY
C         Densmore        15-Dec-82  AUTHOR
C#TYPE    Database low-level interface utility
C#FORMAL PARAMETERS
Cin/out   cursor  cursor index
Cin       flist   DTS field list
Cout      tuple   destination for next tuple
Cout      eof     returns TRUE if no next tuple, FALSE otherwise
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   strngs  string buffers
C#METHOD
C  Calls rdbread
C##
```

AD-A150 423 ALIAS (ACQUISITION AND LOGISTICS INFORMATION AND ANALYSIS SYSTEM) MAINTEN..(U) DECISION-SCIENCE APPLICATIONS INC ARLINGTON VA M S CAREY ET AL. 3/7

UNCLASSIFIED 31 OCT 84 DSA-593-VOL-2 N00014-82-C-0813 F/G 15/5 NL


1.0
4.5
5.0
5.6
6.3
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

```
C     LENIDX*************************************************
$CONTROL segment=dbif,check=2
      SUBROUTINE lenidx(cursor,indx,newidx)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor
      CHARACTER*1 indx(1)
      LOGICAL newidx
C*                                          *** ABSTRACT ***
C#PURPOSE Sets the word length of an index in a cursor
C#AUDIT HISTORY
C       Densmore        22-Mar-83  AUTHOR
C#TYPE  low-level RELATE database utility
C#FORMAL PARAMETERS
Cin     cursor  cursor index
Cin     indx    DTS string describing index...not used
C               if the cursor has no current index
Cin     newidx  .TRUE. if this index was just created
C               and therefore must have length=maxlen
C#COMMON BLOCKS
Cin     cursrs  cursor buffers
C#CALLER  rvc... with sort requests
C#METHOD
C  Lots of RDBINFO calls.
C  Assumes that the index in question is the current index
C  and that indx describes it.
C
C  First, the filenumber and indexnumber are retrieved by
C  an info call using the cursor (current path).  If there is no
C  index then it is assumed crsxl is to be set to the number
C  of words in a tuple.  Otherwise, set crsxl to the number
C  of words in the index described by the fields in indx.
C  Now, the current index has been set using indx, but
C  RELATE is such that there may be MORE fields in the current
C  index.  This occurs whenever an index already exists whose
C  first N fields match the N fields given in indx.  In
C  this case, only the sum of the number of words in the first
C  N fields of the index should be used in setting crsxl.
C#LOCAL VARIABLES
C       info    returned info from RDBINFO (except field numbers)
C       fieldn  returned field numbers from RDBINFO ([1]=quantity)
C       indexn  index number for this path
C       filen   file number for this path
C       maxlen  maximum possible length (words) for index
C       excess  number of extra words given as index length
C               presumably to include the line number field
C       length  length for this index
C       count   number of commas plus one, in indx -- ie.
C                the number of fields in the index actually used
```

```
C          first   first significant character in DTS indx
C          last    last       "            "      "  "    "
C          len     last-first+1
C          char    DO index from first to last
C          i       DO index
C          nfield  fieldn(1) == number of fields in index
C##
```

```
C     RCINIT************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE rcinit
C*                                              *** ABSTRACT ***
C#PURPOSE Initialize RELATE Cursor system for database interfacing
C#AUDIT HISTORY
C        Densmore        17-Dec-82  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#COMMON BLOCKS
Cout      indexs  index buffers
C##
```

```
C     RCKPRV ***************************************************
$CONTROL segment=dbif,check=3
      LOGICAL FUNCTION rckprv(cursor,path)
C*                           *** FORMAL PARAMETER DECLARATIONS ***
      integer cursor
      character*20 path
C*                                                *** ABSTRACT ***
C#PURPOSE   Checks user write privelege on call to a relate
C     utility routine which will change a relation's contents
C#AUDIT HISTORY
C        MSCarey        10-sep-83  AUTHOR
C#FORMAL PARAMETERS
Cin      cursor   relate cursor index
Cin      path     name of the path for this cursor
C#COMMON BLOCKS
Cin      scenar   current scenario information
C#CALLER high-level relate utilities
C#METHOD
C     Write a message if no write privelege and return.
C##
```

```
C     RELCOM **************************************************
$CONTROL check=2,segment=dbif
      SUBROUTINE relcom(incurs,comand)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer incurs
      character comand(1020)
C*                                             *** ABSTRACT ***
C#PURPOSE   Executes a RELATE comand.
C#AUDIT HISTORY
C        MSCAREY        09-aug-83  AUTHOR
C#FORMAL PARAMETERS
Cin      incurs   cursor usage code: 1-mccrs indicates use
C                 specified cursor; >mccrs indicates use any
C                 open cursor.
C#COMMON BLOCKS
Cin      cursrs   relate cursors
Cin      lprnts   debug switches
C#CALLER various
C#METHOD
C     Check the cursor code and set the cursor to use.
C     Extract the comand from the delimited string.
C     Make the call to RELATE, and check for errors.
C#LOCAL VARIABLES
C        !      '    !
C##
```

```
C     RSTIDX*****************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE rstidx(rtn,cursor,relatn,flist)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor
      CHARACTER rtn*6, relatn*255, flist*255
C*                                           *** ABSTRACT ***
C#PURPOSE For RVC sort open routines, SeTs up InDeXes
C#AUDIT HISTORY
C         Densmore        15-Jun-83  AUTHOR
C#TYPE    RELATE Database utility for High-Level routines
C#FORMAL PARAMETERS
Cin       rtn     name of calling routine
Cin       cursor  the cursor just opened by calling routine
Cin       relatn  DTS relation name for the cursor
Cin       flist   DTS field list for the (possibly new) index
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   indexs  index buffers
Cin       lprnts  diagnostic block
C#CALLER  rvcsrt, rvcsts
C#METHOD
C  Calls dcgidx with full fieldlist.  If dcgidx fails, then the
C  desired index is known not to exist; it is created, and the
C  corresponding fieldlist is stored in the idx array.  When
C  dcgidx succeeds it means that the index will not be destroyed
C  when rvclos is called to close the cursor.
C
C  When the fieldlist includes a vertical bar (|) in place of
C  exactly one of the commas (,) delimiting the field names, it
C  means that the caller desires the index to be opened as before,
C  but that only the fields up to the bar are to be used when any
C  calcs are performed.  In this way, one may allow calcs to
C  certain fields, and then guarantee ordered sequential reads
C  for the following fields even though the latter fields are not
C  included in the calc.  This is implemented by searching for the
C  | character and using a different index length (lenidx).
C#LOCAL VARIABLES
C         ibar    location of vertical bar; 0 if none
C         indx    place where idx buffer is located in idx array
C##
```

```
C     RTPADD**************************************************
$CONTROL segment=dbif,check=2
      SUBROUTINE rtpadd(cursor,flist,source)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,source(1)
      CHARACTER*255 flist
C*                                            *** ABSTRACT ***
C#PURPOSE Relate TuPle ADDition: adds tuple to current relation
C#AUDIT HISTORY
C         Densmore        17-Dec-82  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#FORMAL PARAMETERS
Cin       cursor  cursor index for current relation
Cin       flist   field list for tuple
Cin       source  source for new data
C#METHOD
C  Calls DMKTUP
C##
```

```
C     RTPCAL************************************************
$CONTROL segment=dbif,check=2
      SUBROUTINE rtpcal(cursor,keyval,flist,dest,notfnd)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,keyval(1),dest(1)
      CHARACTER*255 flist
      LOGICAL notfnd
C*                                            *** ABSTRACT ***
C#PURPOSE Relate TuPle CALculate: calculates by key-value the next tuple
C         desired from the current relation.
C#AUDIT HISTORY
C         Densmore        17-Dec-82  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#FORMAL PARAMETERS
Cin       cursor  cursor index for the current relation
Cin       keyval  value of the key for the tuple desired; the current
C                  relation must be indexed by this key
Cin       flist   field list for tuple
Cout      dest    output tuple (DESTination)
Cout      notfnd  Logical indicating if the tuple was NOT FouND
C#METHOD
C  Calls dtcalc
C##
```

```
C     RTPDEL************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE rtpdel(cursor)
C*                   *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor
C*                                         *** ABSTRACT ***
C*PURPOSE Relate TuPle DELete: deletes current tuple in current relation
C*AUDIT HISTORY
C         Densmore        17-Dec-82  AUTHOR
C*TYPE    RELATE Database High-Level Interface Utility
C*FORMAL PARAMETERS
Cin       cursor  cursor index for the current relation
C*COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C*METHOD
C  Calls deltup
C**
```

```
C     RTPKIL**************************************************
$CONTROL segment=dbif,check=2
      SUBROUTINE rtpkil(cursor,keyval,notfnd)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,keyval(1)
      LOGICAL notfnd
C*                                                *** ABSTRACT ***
C#PURPOSE finds and deletes (KILls) the tuple whose key is keyval
C#AUDIT HISTORY
C         Densmore        18-Feb-83  AUTHOR
C#TYPE    high-level relate utility
C#FORMAL PARAMETERS
Cin       cursor  cursor index
Cin       keyval  key value -- must correspond to current index
Cin       notfnd  true if NOT FouND
C##
```

201

```
C     RTPNEW**********************************************
$CONTROL segment=dbif,check=2
      INTEGER FUNCTION rtpnew(cursor,flist,source)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,source(1)
      CHARACTER*255 flist
C*                                              *** ABSTRACT ***
C#PURPOSE Relate TuPle NEW-add NEW tuple to curnt relation; returns mode
C#AUDIT HISTORY
C          Densmore        23-Mar-83  AUTHOR
C#TYPE    high level relate DB utility
C#FORMAL PARAMETERS
Cin       cursor  relate cursor
Cin       flist   field list
Cin       source  source data making up tuple
Cfunction rtpnew  0=successful add   1=unary violation
C                 2=EOF--no room to add tuple in file
C##
```

```
C     RTPNFD**************************************************
$CONTROL segment=dbif,check=2
      SUBROUTINE rtpnfd(cursor,keyval,flist,dest,fnd,eof)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,keyval(1),dest(1)
      CHARACTER*255 flist
      LOGICAL fnd,eof
C*                                                  *** ABSTRACT ***
C#PURPOSE Relate TuPle Not Found: calculates by key-value the next tuple
C         desired from the current relation, expecting NOT to match
C         the key value.  Then reads and returns the record
C         which the failed point left us at, i.e. the next
C         greatest value of the key.  Similar to RTPCAL.
C#AUDIT HISTORY
C         MSCarey          31-may-83  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#FORMAL PARAMETERS
Cin       cursor  cursor index for the current relation
Cin       keyval  value of the key for the tuple desired; the current
C                  relation must be indexed by this key
Cin       flist   field list for tuple
Cout      dest    output tuple (DESTination)
Cout      fnd     Logical indicating that an exact match on the
C                 key value was found, which is an error here.
Cout      eof     True if the point left us at the end of the
C                 relation, with no tuple to return.
C#METHOD
C  Calls dtcalc
C##
```

```
C     RTPNXT*************************************************
$CONTROL segment=dbif,check=2
      SUBROUTINE rtpnxt(cursor,flist,dest,eof)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,dest(1)
      CHARACTER*255 flist
      LOGICAL eof
C*                                            *** ABSTRACT ***
C#PURPOSE Relate TuPle NeXT: obtain next tuple in sequence from
C         current relation.
C#AUDIT HISTORY
C         Densmore        17-Dec-82  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#FORMAL PARAMETERS
Cin       cursor  cursor index for current relation
Cin       flist   field list for tuple
Cout      dest    output tuple (DESTination)
Cout      eof     Logical indicating if no more tuples are available
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#METHOD
C  Calls dtnext
C##
```

```
C     RTPREP*************************************************
$CONTROL segment=dbif,check=2
      SUBROUTINE rtprep(cursor,keyval,flist,source,notfnd)
C*                   *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,keyval(1),source(1)
      CHARACTER*255 flist
      LOGICAL notfnd
C*                                             *** ABSTRACT ***
C#PURPOSE finds tuple whose key value is keyval; replaces it w/ source
C#AUDIT HISTORY
C         Densmore        18-Feb-83  AUTHOR
C#TYPE    high-level relate utility
C#FORMAL PARAMETERS
Cin       cursor  cursor index
Cin       keyval  key value -- corresponds to current index
Cin       flist   field list to which tuple source data corresponds
Cin       source  source data for the tuple to be updated
Cout      notfnd  True if tuple was NOT FouND
C##
```

```
C     RTPUPD**********************************************
$CONTROL segment=dbif,check=2
      SUBROUTINE rtpupd(cursor,flist,source)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,source(1)
      CHARACTER*255 flist
C*                                           *** ABSTRACT ***
C#PURPOSE Relate TuPle UPDate: modify the value of the current tuple
C         in the current relation.
C#AUDIT HISTORY
C         Densmore        17-Dec-82  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#FORMAL PARAMETERS
Cin       cursor  cursor index for the current relation
Cin       flist   field list for tuple
Cin       source  source for new tuple data
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#METHOD
C  Calls dcgtup
C##
```

```
C     RVCLOS*************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE rvclos(cursor)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor
C*                                                *** ABSTRACT ***
C#PURPOSE Relate Virtual Cursor CLOSe: close the current relation
C#AUDIT HISTORY
C         Densmore        17-Dec-82  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#FORMAL PARAMETERS
Cin/out   cursor  cursor index for current relation; set to zero
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   indexs  index buffers
C#METHOD
C  Depending on cursor type, calls dcspth or dcslct.
C  If an index was created during opening, it is purged via delidx.
C##
```

```
C     RVCFIL*************************************************
$CONTROL segment=dbif,check=2
      INTEGER FUNCTION rvcfil(relatn,struct)
C*                   *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*255 relatn,struct
C*                                        *** ABSTRACT ***
C#PURPOSE to CREATE a new relation with structure STRUCT
C#AUDIT HISTORY
C         Densmore        22-Feb-83  AUTHOR
C#TYPE    high-level relate utility
C#FORMAL PARAMETERS
Cin       relatn  relation name -- DTS
Cin       struct  the relation structure -- DTS (see RELATE manual)
C#COMMON BLOCKS
Cin/out   cursor  cursor buffers
C##
```

```
C     RVCKIL***********************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE rvckil(cursor,relatn)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor
      CHARACTER*255 relatn
C*                                          *** ABSTRACT ***
C#PURPOSE PURGES the relation named and all its indexes
C#AUDIT HISTORY
C         Densmore       23-Feb-83  AUTHOR
C#TYPE    high-level relate utility
C#FORMAL PARAMETERS
Cin       cursor  cursor index
Cin       relatn  relation name  (file name)
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   indexs  index buffers
Cin       lprnts  diagnostics
C#METHOD
C  cursor must not be from a selection; deallocates index buffers;
C  calls delrel to delete relation and deallocate cursor buffers.
C##
```

```
C     RVCPTH**********************************************
$CONTROL segment=dbif,check=2
      INTEGER FUNCTION rvcpth(relatn,synym,struct)
C*                          *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*255 relatn,synym,struct
C*                                                *** ABSTRACT ***
C#PURPOSE Relate Virtual Cursor PaTH: CREATES a relation under a
C         synonymous name.
C#AUDIT HISTORY
C         Densmore        22-Feb-83  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#FORMAL PARAMETERS
Cin       relatn  Delimited Text String giving relation name
Cin       synym   Delimited Text String giving desired synonym
Cin       struct  DTS structure spec as in RVCFIL
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#METHOD
C  Calls dmkrel and cursor type is noted
C##
```

```
C     RVCREL*************************************************
$CONTROL segment=dbif,check=3
      INTEGER FUNCTION rvcrel(relatn)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*20 relatn
C*                                          *** ABSTRACT ***
C#PURPOSE Relate Virtual Cursor open RELation: opens a relation
C#AUDIT HISTORY
C         Densmore        17-Dec-82  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#FORMAL PARAMETERS
Cin       relatn  Delimited Text String specifying name of relation
Cfunction rvcrel  a cursor index to the new relation
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#METHOD
C  Calls doprel and notes cursor type.
C##
```

```
C     RVCRWD*****************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE rvcrwd(cursor)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor
C*                                           *** ABSTRACT ***
C#PURPOSE Relate Virtual Cursor ReWinD: rewind current relation
C#AUDIT HISTORY
C         Densmore        17-Dec-82  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#FORMAL PARAMETERS
Cin       cursor  cursor index for current relation
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#METHOD
C  calls drewnd.
C##
```

```
C     RVCSLC**************************************************
$CONTROL segment=dbif,check=3
      INTEGER FUNCTION rvcslc(tgtin,unique,keyin,condin)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*255 tgtin,keyin,condin
      LOGICAL unique
C*                                            *** ABSTRACT ***
C#PURPOSE Relate Virtual Cursor SeLeCtion: performs a SELECT operation
C#AUDIT HISTORY
C         Densmore        17-Dec-82  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#FORMAL PARAMETERS
Cin       tgtin   Delimited Text String indicating what fields should
C         TGTLST  be returned, and the values they should assume;
C                 Format: name1[=expr][,name2[=expr]]...
Cin       unique  Logical indicating that selection should result in
C                 unique values in the key list keylst; forces the
C                 specification of keylst
Cin       keyin   Delimited Text String names of fields on which the
C         KEYLST  selection is to be sorted; optional unless unique
C                 is True; avoid specification via the DTS '::'
Cin       condin  Delimited Text String giving the condition under
C         COND    which any virtual tuples created by this select
C                 should be returned as part of the select
Cfunction rvcslc  virtual cursor index pointing to the cursor
C                 associated with the selection results
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#METHOD
C  First, figures out which cursors need to be associated with
C  the new SELECT cursor.  Currently, all open cursors are used.
C  Then, dslect is called to perform the selection command.
C##
```

```
C     RVCSRT************************************************
$CONTROL segment=dbif,check=3
      INTEGER FUNCTION rvcsrt(relatn,flist)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*255 relatn,flist
C*                                          *** ABSTRACT ***
C#PURPOSE Relate Virtual Cursor via SoRT: opens a new relation sorted
C         via a specified key
C#AUDIT HISTORY
C         Densmore        17-Dec-82  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#FORMAL PARAMETERS
Cin       relatn  Delimited Text String naming the relation
Cin       flist   Delimited Text String naming the fields which form
C                 the key upon which to sort
C                 The vertical bar (|) has significance when it
C                 appears in this argument as described in RSTIDX.
Cfunction rvcsrt  cursor index to cursor associated with the named
C                 relation/index pair
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   indexs  index buffers
C#METHOD
C  Calls doppth, then dcgidx.  If dcgidx fails, the index is
C  created via dmkidx, and this fact is noted.
C##
```

```
C     RVCSTS*************************************************
$CONTROL segment=dbif,check=3
      INTEGER FUNCTION rvcsts(relatn,synym,flist)
C*                          *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*255 relatn,synym,flist
C*                                                *** ABSTRACT ***
C#PURPOSE Relate Virtual Cursor via SorT; return Synonym: opens a new
C         relation sorted via a specified key and returns the cursor
C         associated with a synonym to that relation
C#AUDIT HISTORY
C         Densmore         17-Dec-82  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#FORMAL PARAMETERS
Cin       relatn  Delimited Text String naming the relation
Cin       synym   Delimited Text String naming synonym
Cin       flist   Delimited Text String naming the fields which form
C                 the key upon which to sort
C                 The vertical bar (|) character has meaning in
C                 this arg as defined by RSTIDX
Cfunction rvcsts  cursor index to cursor associated with the named
C                 synonym/index pair
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
Cin/out   indexs  index buffers
C#METHOD
C  Calls doppth, then dcgidx.  If dcgidx fails, the index is
C  created via dmkidx, and this fact is noted.
C##
```

```
C     RVCSYN************************************************
$CONTROL segment=dbif,check=3
      INTEGER FUNCTION rvcsyn(relatn,synym)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*255 relatn,synym
C*                                           *** ABSTRACT ***
C#PURPOSE Relate Virtual Cursor SYNonym: opens a relation under a
C         synonymous name.
C#AUDIT HISTORY
C         Densmore         17-Dec-82  AUTHOR
C#TYPE    RELATE Database High-Level Interface Utility
C#FORMAL PARAMETERS
Cin       relatn  Delimited Text String giving relation name
Cin       synym   Delimited Text String giving desired synonym
C#COMMON BLOCKS
Cin/out   cursrs  cursor buffers
C#METHOD
C  Calls doppth and cursor type is noted
C##
```

```
C     RVSCEN ************************************************
$CONTROL segment=dbif,check=3
      SUBROUTINE rvscen(cursor,type,file)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      integer cursor,type
      character*20 file
C*                                              *** ABSTRACT ***
C#PURPOSE   Checks/sets user access flags to the relation
C     being requested, and sets the scenario field key value
C     for this relation for the current scenario.
C#AUDIT HISTORY
C        MSCarey        10-sep-83  AUTHOR
C#FORMAL PARAMETERS
Cin      cursor   utility system cursor index
Cin      type     cursor type (1, 2, or 3)
Cin      file     name of relation to be processed
C#COMMON BLOCKS
Cin      senprm   scenario system parameters
Cin      uzrprv   user privelege information
Cin      snrref   field key values for each relation
Cio      scenar   current scenario settings
C#CALLER relate utilities which open cursors
C#METHOD
C     Set write privelege flag for this cursor according to
C     user priveleges.
C     Make sure the filename has a group suffix.
C     Search snrlnm (extended memory) for a match of the filename
C     On finding a match, set cursen(cursor) to snrlsn(match)
C##
```

10.3 BUILDER-CALLABLE FORTRAN UTILITIES

This Section presents FORTRAN utility routines designed to serve BUILDER screens, with an emphasis on the routines likely to be of interest to developers of any BUILDER-based ALIAS module. All the routines were originally developed to serve the DBU.

Source code for the routines can be found in the slproc.src and sldate.src files. Object code processable by PREP is not maintained; run-time linkable object code is maintained in the account Segmented Library (sl.pub). Update of the contents of sl.pub is accomplished by compiling both source code files into $oldpass, and then running the Segmenter via the "GLUE addsl" command. Note that any new routines must be compiled into the "dsa" segment.

A Segmented Library is the source for program unsatisfied externals at run time. By calling the proper intrinsics, a program can even link routines after execution has begun. BUILDER must do this in order to implement the CALL PROCEDURE facility, since it has no idea what routines might be called until the moment it interprets a CALL PROCEDURE line. Note that routines in an SL may not contain common, data, read, or write statements. Global storage requirements must be met through use of BUILDER memory or extra data segments. For more information about SLs, see the Segmenter manual for the HP 3000.

Table 10-7 contains an annotated listing of all the routines which reside in the SL. The following sections will discuss selected routines by purpose. See Section 10.3.6 for abstracts of the SL routines, which contain detailed calling specifications.

10.3.1 BUILDER-FORTRAN Data Transfer

The BUILDER manual section on CALL PROCEDURE specifies that FORTRAN routines to be called must have three formal parameters: a 50-word integer array for the current cursor, a "table" of

Table 10-7. BUILDER-Callable Routines in the SL

ROUTINE	PURPOSE
ABTRNS	Aborts a transaction on all son RELATE son processes started up via the file management subsystem. See the RELATE reference manual for a discussion of what transactions are.
BGTRNS	Like ABTRNS, but a global BEGIN TRANSACTION.
CALCDATE	Specialty routine serving the PROJ_NC_SKED DBU screen. Allows quick recalculation of ship schedule dates given a basis date and a set of planning factors (intervals between milestones).
CDTODD	Character Date TO DDate. Same as utility of same name in RL. In SL only to allow use by BUILDER-called routines.
CURINI CURSWP	These two routines form the FORTRAN part of the multiple-RELATE-son-process relation management system. This system is usable from any BUILDER module, not just the DBU. The only restriction is that each user of the system must specify a unique value of the SCREENSYS Job Control Word. See Section 8.4.3.2 for further detail on the system. CURINI initializes a file management system invocation, CURSWP swaps a cursor from the system's storage in an extra data segment into a BUILDER partition.
DATEMK DCLRFY DDTOCD DDTOID	Date utility routines virtually identical to their counterparts of the same names in UTLR (see Section 10.1). In SL to allow usage by BUILDER-called routines.
DLTRIM	Similar to ltrim in UTLR. Integer function returning the leftmost non-blank character of a string.
DOTRNS	Similar to ABTRNS above, but does a global COMMIT TRANSACTION command.
DRTRIM	Similar to rtrim in UTLR. Integer function returning the rightmost non-blank character in a string.
DRUNED DRUNTDP	These routines create son processes running the HP and TDP editors, respectively. They are obsolete now that BUILDER can transparently create son processes for you when you give :RUN commands.
DSAFETCH	Moves a string from a specified word address into an integer array. Useful when the CALL PROCEDURE needs to be read using the contents of the third formal parameter.

Table 10-7. BUILDER-Callable Routines in the SL

ROUTINE	PURPOSE
DSAGETC	Same as dsafetch, but transfers string from a byte address into a character array.
DSAPUT	Transfers a character string value to a given address in BUILDER memory.
GETSCENV	This routines takes a relation name and a screen variable name from the CALL PROCEDURE line, looks up the relation and its associated current scenario key field value in the scenario system extra data segment, and puts this value into the given screen variable. The routine is necessary to enforcement of scenario security in any BUILDER-based module.
GETVAR	Retrieves the (ASCII) contents of a screen variable specified by name and places them in a character variable.
IDTODD LMONTH MODCOR MRKDAY NWDATU NWIDAT	More date routines identical to their UTLR counterparts. These duplicates are here in the SL so they can be called by the BUILDER-called routines. See Section 10.1 for a description of each.
PREPREPT	A pre-processor for RELATE EXECUTE files giving authors of such files the capability to enforce scenario security. Preprept opens and reads a file named on its CALL PROCEDURE line, echoing the file's records to a temporary file. It searches each record for instances of "[relation.group]", looks up each such relation name found in the scenario system extra data segment, and substitutes the appropriate key value between (and including) the brackets. Thus selections can be given to limit the data returned to that of a particular scenario, without know <u>which</u> scenario in advance.
PUTVAR	Like getvar, but writes the contents of a FORTRAN character variable to the address of a screen variable specified by name.
SPSUSP	Suspends the current BUILDER process and activates its father. Useful for any module which it is desirable to put on "hold" (as opposed to termination by the BUILDER EXIT command) when the user returns to the command system, as the DBU is.

unspecified length which contains pointers into the BUILDER memory map, and an array of addresses and lengths which allow access to the interpreted text of the CALL PROCEDURE line.

In order for a BUILDER-called FORTRAN routine to be truly useful, there must be a means for passing data between the screen and the routine. The transfer can be done by file/relation i/o, but this is clumsy. Much more convenient is copying of data between BUILDER variables and variables local to the routine.

The PUTVAR and GETVAR utilities make use of the information in the second formal parameter to implement such a capability. The author of the FORTRAN routine need only know the name(s) of the screen variable(s) to/from which data is to be transferred. Note that BUILDER stores all data in an ASCII format, regardless of the type declaration in the screen, so type conversion will be necessary within the routine for numeric data. See the abstracts of these routines for specifics about the table of pointers into the BUILDER memory map if you are interested in that.

A more primitive but occasionally useful capability is provided by accessing the text of the CALL PROCEDURE line. This can be done by proper use of the DSARTRIM routine. DRTRIM and DLTRIM are useful in parsing the string extracted through use of dsartrim. See the code of the preprept or getscenv routines for examples of how this is done.

10.3.2 Scenario Security Enforcement Assistance

One of the most serious problems facing the designer of a BUILDER-based ALIAS module is the matter of scenario security. Like any other module, these must not access or change data for scenarios other than the user's current one, and no modifications can be allowed to data in relations that have only indirect access status for the given scenario.

In a FORTRAN module use of the DBIF to open and operate on relations automatically places the proper scenario field key value for each open relation in the /scenar/ common block, making it fairly easy to construct selections or point/read strategies which return only the proper data. However, none of these facilities are available from BUILDER.

The GETSCENV and PREPREPT routines solve these problems by extracting scenario field key values from the scenario system extra data segment in the same way that DBIF routines do.

Getscenv takes a relation name and a screen variable name and returns the scenario field key value for that relation for the current scenario into the screen variable. This allows the screen designer to construct selects and point/read strategies in a way that maintains scenario security.

Designers of reports in the form of RELATE EXECUTE files, run from BUILDER, are able to enforce scenario security by use of the PREPREPT routine. The problem again is to construct a WHERE clause of the form WHERE SCENARIO="key_value", where the key_value is the one appropriate to the given relation and scenario. When preprept is available, this can be done reliably by substituting the phrase [relation.group] for key_value. Given the EXECUTE file name, preprept will read the file, echoing to a temporary named DBURTEMP. Whenever it encounters a relation name in brackets, it will search the scenario system extra data segment for the key_value currently appropriate for that relation, and will substitute that value for the bracketed expression. The screen can then EXECUTE DBURTEMP.

10.3.3 File Management

The DBU file management subsystem was discussed in detail in Section 8.4.3.2. It allows a BUILDER-based module to use a large number of relations simultaneously by operating multiple RELATE son processes. The subsystem was designed in a manner that allows it to be used by several concurrently existing BUILDER processes, the only restriction being that each use a unique value for the SCREENSYS Job Control Word.

The subsystem consists of the CURINI and CURSWP routines for relation/partition management, and the ABTRNS, BGTRNS, and DOTRNS routines for global transaction management.

10.3.4 Process Handling

BUILDER-based modules which it is desirable to have exist permanently (in a suspended state) when the user is exercising a different part of the system may call the SPSUSP routine to suspend themselves without terminating. Note that the process creation/activation logic must also be properly arranged in the mrunp routine.

10.3.5 Other Capabilities

A quite DBU-specific utility, CALCDATE, was created to perform schedule date recalculations for the PROJ_NC_SKED screen. Although unlikely to be of use to other modules, a large number of date utility routines were duplicated (from UTLR) to serve calcdate. These may prove useful to screens with heavy date-processing requirements.

10.3.6 Abstracts for SL Routines

Only abstracts for routines not duplicated from UTLR appear in this section. See Section 10.1 for descriptions of the date utilities (and (d)rtrim and (d)ltrisa) found in sl.pub.

```
C     ABTRNS ***********************************************
$control segment=dsa
      SUBROUTINE  abtrns(cursor,table,pointr)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer cursor(50),table(1),pointr(4)
C*                                            *** ABSTRACT ***
C#PURPOSE   Does a relate ABORT TRANSACTION on all active
C     RELATE son processes EXCEPT that started up by the builder.
C#AUDIT HISTORY
C        MSCarey        09-jan-83  AUTHOR
C#FORMAL PARAMETERS
Cin      all      arguments from builder CALL PROCEDURE facility
C#COMMON BLOCKS
C        none
C#CALLER DBU via CALL PROCEDURE
C#METHOD
C     Retrieve the cursor storage data segment, which contains
C     ids of all active relate son processes in words 51-100, and
C     pointers to the locations of cursors open on each of these
C     processes in words 101-150.  In each case, a value of 0 terminates
C     the list of data for active sons.  Retrieves the info for each of
C     these cursors in turn and gives the ABORT TRANSACTION command.
C#LOCAL VARIABLES
C        procs    son process ids (1-50), pointers to cursors (51-100)
C##
```

```
C     BGTRNS *************************************************
$control segment=dsa
      SUBROUTINE  bgtrns(cursor,table,pointr)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
      integer cursor(50),table(1),pointr(4)
C*                                              *** ABSTRACT ***
C#PURPOSE   Does a relate BEGIN TRANSACTION on all active
C     RELATE son processes EXCEPT that started up by the builder.
C#AUDIT HISTORY
C        MSCarey        09-jan-83  AUTHOR
C#FORMAL PARAMETERS
Cin      all       arguments from builder CALL PROCEDURE facility
C#COMMON BLOCKS
C        none
C#CALLER DBU via CALL PROCEDURE
C#METHOD
C     Retrieve the cursor storage data segment, which contains
C     ids of all active relate son processes in words 51-100, and
C     pointers to the locations of cursors open on each of these
C     processes in words 101-150.  In each case, a value of 0 terminates
C     the list of data for active sons.  Retrieves the info for each of
C     these cursors in turn and gives the BEGIN TRANSACTION command.
C#LOCAL VARIABLES
C        procs    son process ids (1-50), pointers to cursors (51-100)
C##
```

```
C     CALCDATE ******************************************************
$CONTROL segment=dsa
      SUBROUTINE  calcdate(cursor,table,pointr)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      integer cursor(50),table(41),pointr(4)
C*                                                  *** ABSTRACT ***
C#PURPOSE   Implements the ESC R function for projected
C     new construction schedules DBU screen; recalculates
C     schedule dates using planning factors.
C#AUDIT HISTORY
C         MSCarey        02-apr-84  AUTHOR
C#FORMAL PARAMETERS
Cin       cursor   current DBU cursor
Cin       table    primary builder memory table
Cin       pointr   pointers to call procedure line text
C#COMMON BLOCKS
C         none
C#CALLER DBU, screen PROJ_NC_SKED
C#METHOD
C     Get the test from the call procedure line; first argument is
C     name of variable which is basis date; second argument is basis
C     date.
C     Load planning factors from builder memory.
C     Call ascdaysl for each date.
C     Place the new date in the proper variable in builder memory.
C#LOCAL VARIABLES
C         maxdat   max number of schedule dates processable
C         basind   index number of the basis date
C         basnam   name of the basis date
C         basdat   value of the basis date, ddate format
C         timunt   time units specification
C         dlist    list of dates to be processed, in ascending order
C         plist    list of planning factors, in ascending order
C         day      ddate representation of each new date
C         datnam   name of each date to process
C##
```

```
C     CURINI ****************************************
$CONTROL segment=dsa,uslinit
      SUBROUTINE curini(cursor,table,pointr)
      integer cursor(50),table(1),pointr(4)
C*                                              *** ABSTRACT ***
C#PURPOSE   Initializes a set of cursors for use by the data
C     entry system.  Works in concert with CURSWP to.  Allows use
C     of multiple RELATE processes as sons of the builder.
C     Compiled code resides in SL.PUB
C#AUDIT HISTORY
C        MSCarey         30-sep-83  AUTHOR
C#FORMAL PARAMETERS
Cin      cursor   cursor array used by screen system
Cin      table    global data storage table for screen system
Cin      pointr   pointers to argument from call
C#COMMON BLOCKS
C        none
C        data segment format is:
C           location 0-9   index of cursor now in use by system
C                          by builder USE cursor id number
C           location 50-10000 by 50's: cursor data arrays
C#CALLER CRI builder application files
C#METHOD
C     Routines resident in an SL may not have global data
C     declarations.  The screen system multiple cursor facility
C     simulates global storage for the cursors by using an
C     extra data segment.  This routine initializes that data
C     segment.
C
C     The id of the data segment WAS taken from the argument supplied
C     on the CALL PROCEDURE line in the application file.  This
C     argument must be numeric >0 and <32767.
C     A bug in the builder now prevents this.  JOB CONTROL WORDS
C     are currently used to communicate the id number of the cursor
C     desired, the id number of the USE cursor to be swapped into, and
C     the id number of the son process to use.
C
C     The routine does not actually initialize any cursors; this is
C     done by CURSWP when it detects a 50-word data segment area
C     which is not yet initialized.  This routine writes codes into
C     a word of each cursor area which tell CURSWP that no
C     rcinitx call has yet been done.  A 0 is placed in word 48,
C     which RELATE uses to store son process id's in.  This word
C     will never be 0 once rcinitx has been called for a cursor.
C#LOCAL VARIABLES
C        numcur   number of cursors usable by system
C        newcur,lcurs   array of cursors to be initialized and stored
C        inuse    cursor currently in use by system
C        iarg,carg,arg  argument from call in various forms
```

```
C                     remaining arguments are for intrinsic calls
C**
```

```
C     CURSWP ****************************************************
$CONTROL segment=dsa
      SUBROUTINE curswp(cursor,table,pointr)
C*                      *** FORMAL PR DECLARATIONS ***
      integer cursor(50),table(1),pointr(4)
C*                                              *** ABSTRACT ***
C#PURPOSE   Swaps the cursor currently in use by the screen
C     system into cursor memory and brings in the cursor
C     requested in the argument attached to the CALL PROCEDURE call
C     to this routine (NOW READS JCW).  See CURINI.  Compiled code
C     resides in SL.PUB
C#AUDIT HISTORY
C        MSCarey        30-sep-83  AUTHOR
C#FORMAL PARAMETERS
C      Currently in use by screen system
Cin      table    screen system global memory table
Cin      pointr   pointers to argument of CALL PROCEDURE
C#COMMON BLOCKS
C        none
C#CALLER CRI builder application files
C#METHOD
C no  Parse the argument, whose format is D.C, where D is the
C no  id of the data segment specified in a dsacursorinit call,
C no  and C is the index of the cursor which the application wishes
C no  swapped in for its use.
C
C     Retrieve the SCREENSYS, NUMSWAP, CURSORNUM, AND CURSORPROC
C     Job Control Words, which specify the cursor memory data
C     segment id, the id of the builder USE cursor to be swapped,
C     the id number of the cursor to be swapped into 'numswap',
C     and the son process id code to be given to rdbinitx if
C     'cursornum' is not yet initialized.
C
C     Get the index of the data segment, swap out the current cursor,
C     and swap in the one desired.
C##
```

```
C     DOTRNS ***********************************************
$control segment=dsa
      SUBROUTINE  dotrns(cursor,table,pointr)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      integer cursor(50),table(1),pointr(4)
C*                                                *** ABSTRACT ***
C#PURPOSE   Does a relate COMMIT TRANSACTION on all active
C     RELATE son processes EXCEPT that started up by the builder.
C#AUDIT HISTORY
C        MSCarey        09-jan-83  AUTHOR
C#FORMAL PARAMETERS
Cin      all       arguments from builder CALL PROCEDURE facility
C#COMMON BLOCKS
C        none
C#CALLER DBU via CALL PROCEDURE
C#METHOD
C     Retrieve the cursor storage data segment, which contains
C     ids of all active relate son processes in words 51-100, and
C     pointers to the locations of cursors open on each of these
C     processes in words 101-150.  In each case, a value of 0 terminates
C     the list of data for active sons.  Retrieves the info for each of
C     these cursors in turn and gives the COMMIT TRANSACTION command.
C#LOCAL VARIABLES
C        procs    son process ids (1-50), pointers to cursors (51-100)
C##
```

```
C     DRUNED ***********************************************
$CONTROL segment=dsa
      SUBROUTINE  druned(cursor,table,pointr)
C*                          *** FORMAL PARAMETER DECLARATIONS ***
      integer cursor(50),table(1),pointr(4)
C*                                               *** ABSTRACT ***
C#PURPOSE   Runs HP editor as a son of the screen system.
C#AUDIT HISTORY
C        MSCarey        25-nov-83  AUTHOR
C#FORMAL PARAMETERS
Cin      cursor   relate cursor in use at time of call
Cin      table    screen system io table
Cin      pointr   pointers to call parameter info
C#CALLER BUILDER procedure
C#METHOD
C     Call to system intrinsic CREATE
C#LOCAL VARIABLES
C        pin      son process id number
C        flag     argument to create; value of 1 causes screen
C                 reactivation when son teminates.
C##
```

```
C      DRUNTDP ************************************************
$CONTROL segment=dsa
      SUBROUTINE  druntdp(cursor,table,pointr)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      integer cursor(50),table(1),pointr(4)
C*                                                *** ABSTRACT ***
C#PURPOSE   Runs TDP editor as a son of the screen system.
C#AUDIT HISTORY
C         MSCarey        25-nov-83  AUTHOR
C#FORMAL PARAMETERS
Cin       cursor   relate cursor in use at time of call
Cin       table    screen system io table
Cin       pointr   pointers to call parameter info
C#CALLER BUILDER procedure
C#METHOD
C      Call to system intrinsic CREATE
C#LOCAL VARIABLES
C         pin      son process id number
C         flag     argument to create; value of 1 causes screen
C                  reactivation when son teminates.
C##
```

```
C     DSAFETCH*********************************************
$CONTROL segment=dsa,check=0
      SUBROUTINE dsafetch(data,datalen,address,length)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer datalen,data(datalen),address(1),length
C*                                          *** ABSTRACT ***
C#PURPOSE   Converts the pointer information provided by the
C     CRI screen application builder CALL PROCEDURE facility
C     into an integer argument string usable by fortran.
C#AUDIT HISTORY
C        MSCarey        28-sep-83  AUTHOR
C#FORMAL PARAMETERS
Cout     data     argument string, integer form
Cin      datalen  max length of string in words
Cin      address  array mapped onto stack location where
C                 argument string is stored
Cin      length   length of argument string in bytes
C#COMMON BLOCKS
C        none
C#CALLER dsapoint,dsacursorinit,dsausecursor
C#METHOD
C     Calling routines provide a target array (data) which this
C     routine transfers the argument into.
C     Calling routines receive the word address in the stack of the
C     argument data in an integer word.  By denoting this as a call-
C     by-value argument (syntax  ) in the calling routine,
C     while causing dsafetch to think it is a normal call-by-
C     reference, the address array in this routine is mapped onto
C     the proper location in the stack.
C##
```

```
C     DSAGETC ***********************************************
$CONTROL segment=dsa,check=0
      SUBROUTINE dsagetc(data,address,length)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      character*1 data(length),address(length)
      integer length
C*                                              *** ABSTRACT ***
C#PURPOSE   Converts the pointer information provided by the
C     CRI screen application builder CALL PROCEDURE facility
C     into a character argument string usable by fortran.
C#AUDIT HISTORY
C        MSCarey        28-sep-83  AUTHOR
C#FORMAL PARAMETERS
Cout     data     argument string, character form
Cin      address  array mapped onto stack location where
C                 argument string is stored
Cin      length   length of argument string in bytes
C#COMMON BLOCKS
C        none
C#CALLER dsapoint,dsacursorinit,dsausecursor
C#METHOD
C     Calling routines provide a target array (data) which this
C     routine transfers the argument into.
C     Calling routines receive the byte address in the stack of the
C     argument data in a character word.  By denoting this as a call-
C     by-value argument (syntax ) in the calling routine,
C     while causing dsagetc to think it is a normal call-by-
C     reference, the address array in this routine is mapped onto
C     the proper location in the stack.
C##
```

```
C     DSAPUT*********************************************
$CONTROL segment=dsa,check=0
      SUBROUTINE dsaput(data,datalen,address,length)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      integer datalen,data(datalen),address(datalen),length
C*                                          *** ABSTRACT ***
C#PURPOSE   Writes 'data' to the given address in builder memory.
C#AUDIT HISTORY
C        MSCarey        28-sep-83  AUTHOR
C#FORMAL PARAMETERS
Cout     data     argument string, integer form
Cin      datalen  max length of string in words
Cin      address  array mapped onto stack location where
C                 argument string is stored
Cin      length   length of argument string in bytes
C#COMMON BLOCKS
C        none
C#CALLER dsapoint,dsacursorinit,dsausecursor
C#METHOD
C     Calling routines provide a target array (data) and an address.
C     The call-by-value/check=0 trick is used to make this routine
C     see the address as a fortran array into which it can write.
C     See also routine dsafetch.
C##
```

```
C     GETSCENV ***********************************************
$CONTROL segment=dsa
      SUBROUTINE  getscenv(cursor,table,pointr)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer cursor(50),table(1),pointr(4)
C*                                          *** ABSTRACT ***
C#PURPOSE   Takes a DB relation name and a target variable
C     name as input, finds the scenario field value for that
C     relation for the current scenario, and places the value
C     in the target variable.
C#AUDIT HISTORY
C        MSCarey        17-mar-84  AUTHOR
C#FORMAL PARAMETERS
Cin      cursor   current builder cursor
Cin      table    builder memory map
Cin      args     pointer to file name to process
C#COMMON BLOCKS
C        none
C#CALLER builder procedures
C#METHOD
C     Use dsafetch to get the argument text.
C     Get the file name from the argument string.
C     Look through the scenario extra data segment and get the
C      proper field value.
C     Get the target variable name.
C     Use putvar to place the value into the given variable in
C     builder memory.
C#LOCAL VARIABLES
C        filnam   name of report template file, with group suffix
C        com      command/filename string
C        rec      record read from input file
C        sname    file name read from scenario data segment
C        scen     scenario field value for given file
C        name     name of target file, as parsed from input record
C        filin,filout   MPE file numbers for input and output files
C##
```

```
C     GETVAR ************************************************
$CONTROL segment=dsa
      SUBROUTINE  getvar(table,varnam,value,valdim,len)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      integer valdim,table(41),len
      character*16 varnam,value*(valdim)
C*                                          *** ABSTRACT ***
C#PURPOSE   Retrieves the value of a variable in builder memory.
C#AUDIT HISTORY
C        MSCarey        02-apr-84  AUTHOR
C#FORMAL PARAMETERS
Cin      table    builder memory map start
Cin      varnam   name of variable to look for
Cout     value    value of variable found
Cout     len      length of variable's storage area
C#COMMON BLOCKS
C        none
C#CALLER builder procedures
C#METHOD
C     see routine putvar
C##
```

```
C     PREPREPT *****************************************************
$CONTROL segment=dsa
      SUBROUTINE  preprept(cursor,table,pointr)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
      integer cursor(50),table(1),pointr(4)
C*                                                 *** ABSTRACT ***
C#PURPOSE   Takes a file name, opens and reads the file,
C        substituting proper scenario field values for every
C        occurrence of {file.group}, and writes the result to
C        temporary file DBURTEMP, overwriting it if there.
C        Preprocessor for RELATE report generation files, making
C        them produce output for the proper scenario.
C        This version serves the BUILDER.
C#AUDIT HISTORY
C        MSCarey        17-mar-84  AUTHOR
C#FORMAL PARAMETERS
Cin      cursor    current builder cursor
Cin      table     builder memory map
Cin      args      pointer to file name to process
C#COMMON BLOCKS
C        none
C#CALLER builder procedures
C#METHOD
C     Use dsafetch to get the file name text.  Do the file opens,
C     quitting if can't find input file, and then process.
C#LOCAL VARIABLES
C        filnam    name of report template file, with group suffix
C        com       command/filename string
C        rec       record read from input file
C        sname     file name read from scenario data segment
C        scen      scenario field value for given file
C        name      name of target file, as parsed from input record
C        filin,filout   MPE file numbers for input and output files
C##
```

```
C     PUTVAR **************************************************
$CONTROL segment=dsa
      SUBROUTINE putvar(table,varnam,value,len)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer table(41),len
      character*16 varnam
      character*(len) value
C*                                            *** ABSTRACT ***
C#PURPOSE   Finds the given variable location in builder memory
C     and writes the given value there.
C#AUDIT HISTORY
C         MSCarey        19mar-84  AUTHOR
C#FORMAL PARAMETERS
Cin       table    map to builder memory
Cin       varnam   name of variable in builder application
Cin       value    new value for varnam
Cin       len      length of new value
C#COMMON BLOCKS
C         none
C#CALLER CRI builder
C#METHOD
C     table(41) is word address of start of variable table, which
C     is a linked list with 19-word elements.  Elements of interest
C     are (1): word address next cell; (10): byte pointer to fieldname
C     (2): length of field name; (9) word address of data area;
C     (19) length of data area in bytes; (3) len data words
C
C     liberal use of dsafetch.  map vartab onto variable table, then
C     run down the linked list looking for a match on the variable
C     provided.  When found, map ival onto its data area and set
C     to new value.
C#LOCAL VARIABLES
C         vartab   a 19-word builder variable linked list cell
C         ifld,fldnam variable name as taken from builder memory
C         valbuf,valchr  word-aligned buffer for new value
C##
```

```
C     SPSUSP ***********************************************
$CONTROL segment=dsa
      SUBROUTINE spsusp(cursor,table,pointr)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      integer cursor(50),table(1),pointr(4)
C*                                          *** ABSTRACT ***
C#PURPOSE   When called from a screen application file,
C     suspends operation of the application rather than
C     terminating its execution.  Thus next call to application
C     avoids initialization work.  When application is not a
C     son process, merely causes an abort.
C#AUDIT HISTORY
C        MSCarey        30-sep-83  AUTHOR
C#FORMAL PARAMETERS
Cin      cursor   current screen system cursor
Cin      table    screen system memory
Cin      pointr   pointers to arguments on CALL PROCEDURE line
C#COMMON BLOCKS
C        none
C#CALLER screen application files
C#METHOD
C     Activates father process and suspends this one.
C##
```

10.4 BUILDER UTILITY SCREENS

This section will present a few BUILDER subroutine screens which can be thought of as utilities by the generality of their purpose and their limited data structure requirements. The screens are MPECOMMAND, RUNEDITOR, RUNTDP, and SEARCH. They are currently used solely by the DBU; their code resides in the dbusubr.screens file.

The selection of these few screens as utilities is somewhat arbitrary---there are many more screens in the DBU targeted toward performance of specific tasks, and extractable for use by other screen systems. Examples are the various command processing utilities and the comment screen support utilities handling text data. Developers of large screen systems should peruse Section 8.4 both for other screens which might be useful and for general approaches BUILDER procedure design and implementation.

The screens are displayed in Figures 10-1 through 10-4.

The MPECOMMAND screen simulates the monitor by entering a loop in which the user is prompted with the standard ":" for commands. BUILDER is instructed to treat the user's input as an MPE command request by a line of the form ":%mpecom". MPE command execution from within a system is a nice feature, but its usefulness is somewhat limited by the fact that UDC's cannot be executed this way. The screen has a minimal data structure requirement: a single string buffer called MPECOM (should be at least 80 characters long, better 132).

The RUNEDITOR and RUNTDP commands bring up the HP standard editor and the TDP editor as son processes, automatically doing a SYSTEM $CANCEL to reset the control-Y trap and prompting for a screen rewrite (REFRESH must be given in the caller) when the editor returns control to BUILDER. The routines require two alphanumeric variables, Y and COMMAND.

Figure 10-1. MPECOMMAND Utility Screen

```
*** SCREEN MPECOMMAND
*** INITIAL
    SCROLL "Please give an MPE command at the colon (no UDCs) or
            RETURN to do nothing."
    MPECOM := ""
    PROMPT ":",MPECOM
    WHILE MPECOM  ""
    IGNORE ALL ERRORS
    :%mpecom
    IF $ERROR
        DISPLAY "%$errmsg"
        ENDIF
    MPECOM := ""
    PROMPT ":",MPECOM
    ENDWHILE
    SYSTEM $CANCEL
    PROMPT "Hit RETURN to refresh screen.",MPECOM
    COMMAND := ""
    RETURN SCREEN
```

Figure 10-2. RUNTDP Utility Screen

```
*** SCREEN RUNTDP
*** INITIAL
    SCROLL "Entering tdp editor..."
    SCROLL " "
    :RUN TDP.PUB.SYS
    SYSTEM $CANCEL
    PROMPT "Hit RETURN to refresh screen ",Y
    COMMAND := ""
    RETURN SCREEN
```

Figure 10-3. RUNEDITOR Utility Screen

```
*** SCREEN RUNEDITOR
*** INITIAL
    SCROLL "Entering HP editor..."
    SCROLL " "
    :RUN EDITOR.PUB.SYS
    SYSTEM $CANCEL
    PROMPT "Hit RETURN to refresh screen ",Y
    COMMAND := ""
    RETURN SCREEN
```

Figure 10-4. SEARCH Utility Screen

```
*** SCREEN SEARCH
*** INITIAL
    COND := ""
    NUM := $RDBINFO(200)
    FLDNUM := 1
    WHILE FLDNUM = NUM
         LIST := $RDBINFO(201,FLDNUM)
         FLDNAM := $ITEM(LIST,1,1)
         IGNORE ERROR 7313
         FLDVAL := %fldnam
         IF (NOT $ERROR) AND FLDVAL  "" AND FLDVAL  "0"
              FLDTYP := $ITEM(LIST,3)
              FLDFMT := $ITEM(LIST,11)
              FLDLEN := $ITEM(LIST,5)
              IF FLDTYP=1
                   COND := $CONCAT(COND," %fldnam","="," ""%fldval"" "," AND")
                   FLDVAL := $CONCAT(FLDVAL,$SUBSTR(ZEES,1,FLDLEN
                                       -$LENGTH(FLDVAL) ))
                   COND := $CONCAT(COND," %fldnam","="," ""%fldval"" "," AND")
              ELSEIF FLDTYP=4 AND $BITS(FLDFMT,13,3)=1
                   COND := $CONCAT(COND," %fldnam","="," ""%fldval"" "," AND ")
              ELSE
                   COND := $CONCAT(COND, ' %fldnam","=",FLDVAL," AND ")
```

Figure 10-4. SEARCH Utility Screen

```
            ENDIF
        ENDIF
        FLDNUM := FLDNUM+1
ENDWHILE
IF COND=""
        DISPLAY "You must fill in the field values you wish a match
                found for."
      OK := 0
        RETURN SCREEN
        ENDIF
COND := $SUBSTR(COND,1,$LENGTH(COND)-4)
SET OPTION QUOTES=NO
SELECT  BY %nowindex WHERE %cond %modeset
SET OPTION QUOTES=YES
IF NOT $FOUND
        DISPLAY "No match found."
        OK := 0
ELSE
        DISPLAY
        OK := 1
ENDIF
RETURN SCREEN
```

The SEARCH screen implements the DBU "S" command. It constructs a selection on the current partition which requires that records returned match all non-blank values of screen variables which match field names on the current path. This allows users to immediately jump to a record in a file if they know enough about the record to uniquely identify it.

A nice feature of the screen is its handling of alpha variables. Instead of requiring an exact match on these, the constructed selection allows a range of values spanning any trailing blanks in the value in the BUILDER variable. For example, if a DBU user wanted all schedule records for ships with class names beginning with "D", he could place "D" in the class field and give the "S" command. The resulting selection's WHERE clause would read WHERE SCENARIO="%scenario" AND CLASS>="D" AND CLASS<="Dzzzzzzzzz", which would produced the desired effect.

The screen's logic is based on a loop which builds up the WHERE clause by concatenation into a buffer. The loop runs over all fields on the current path. The number, names, and data types of the fields are discovered by $RDBINFO calls. Data type is important because string and date values must be delimited by quotes, while numbers must be undelimited. Phrases are added to the clause only for fields which have a non-blank value in the screen variable with the same name (an error results on the attempt to assign the field's value to the buffer FLDVAL if there is no such variable). If all the relevant screen variables are blank then COND will be blank and the routine will print an error message and return. Otherwise the selection is given and the OK flag variable is set to communicate whether at least one matching record was found to the caller. It is up to the caller to retrieve the variable.

SEARCH requires the COND, NUM, FLDNUM, LIST, FLDNAM, FLDVAL,FLDTYP, FLDFMT, FLDLEN, MODESET, NOW_INDEX, and OK variables already exist before it is called. Note that the BY clause to be used in the selection and any hard-wired WHERE clause conditions must be placed in the NOW_INDEX and MODESET variables before SEARCH is called.

10.5 FORTRAN INCLUDE FILES

This Section presents the code of all ALIAS FORTRAN include files. These files, stored in the .incl group, contain ALIAS common block declarations. They also include some "parameter" blocks, i.e. sets of FORTRAN parameter statements which specify array dimensions and other static system information, and some DATA statements, particularly field lists for relations.

The source code for these global data structures is maintained in files separate from the source of routines to promote standardization and maintainability. See the discussions of the include methodology in Sections 2 and 6 for further information about this.

The include files are presented on the following pages in alphabetical order, often more that one to a page in order to conserve space. Note that their source code was current as of September, 1984.

10.6 MISCELLANEOUS SYSTEM RESOURCES AND THEIR CURRENT UTILIZATION

Resources which are difficult to categorize, but which are nevertheless very important to software developers, include locations in the /lprnts/ FORTRAN common block and usage of extra data segments.

Lprnts array locations are typically reserved for one routine or a few related routines. This way the diagnostic output that results from turning on a given lprnt is precisely targeted. Table 10-8 presents a list of the current usage of lprnts locations.

FILE ADDSUB

```
C
C  INCLUDE FILE ADDSUB
C  ---common blocks for the ADDSUB environment utility program
C  ---which supports maintenance of entries.doc and subkeys.doc
      COMMON /entrys/ module,source,object,entryn,subprg,
     1 caltyp,prpose,author,cat,mkey
      CHARACTER module*12,source*18,object*18,entryn*12,
     1 subprg*12,caltyp*10,prpose*48,author*12,cat*12,mkey*12
      PARAMETER elen=166
      PARAMETER eflist=':MODULE,SOURCEFILE,OBJECTFILE,ENTRYNAME,SUBPROGR
     1AM,CALLTYPE,PURPOSE,AUTHOR,CATEGORY,MAINKEY:'
      PARAMETER efile=':ENTRIES.DOC:'
      INTEGER ientry(1)
      EQUIVALENCE (module,ientry)
C
      COMMON /subkey/ entrys,keywrd
      CHARACTER*12 entrys,keywrd
      PARAMETER slen=24
      PARAMETER sflist=':ENTRY,KEYWORD:'
      PARAMETER sfile = ':SUBKEYS.DOC:'
      INTEGER isubk(1)
      EQUIVALENCE (entrys,isubk)
C
      COMMON /escrs/ ecrs,scrs,dauth
      INTEGER ecrs,scrs
      CHARACTER*12 dauth
C
```

FILE ASGN

```
C
C  include file /ASGN/ -- MANUAL ASSIGNER DATA BLOCKS
                          SEE FILE ASGNDESC.INCL FOR DOCUMENTATION
C *** PARAMETER lnasgn=30+maxper+2*maxyds, lbasgn=2+6+2*maxper
      PARAMETER mperh =  20, maxyds= 99, maxper= 208, lcasgn= 1240,
     1          lnasgn= 436, lbasgn=426, cundef=-9999,
     2          zdel  = %5C, mcyds = 12, mccls =  12, mccls1=   11
      PARAMETER qpfx=4,qcls=5,qyrd=6,qjob=7
C
      COMMON /casgn/ scname,perlbl,duratn,cpagep,spromt(2),apromt(16),
     1          ydname(maxyds)
      CHARACTER scname*16,   perlbl*8,duratn*8,cpagep*2,cpageh*1,
     1          ydname*mcyds,spromt*1,apromt*1
      INTEGER casgn(1)
      EQUIVALENCE (cpagep,cpageh), (scname,casgn)
C
      COMMON /nasgn/ idurat,ordmod,fidate,npagev,npageh,npromt,numper,
     1        numyds,asntot,grdtot,lmore ,topyd ,lowyd ,topidx,lowidx,
     2        freptr,recalc,ydcalc,mpageh,prompt,fiyear,
     3        nvruse,asnsav,asncas,kpad(6),
     4        sumper(maxper),firstp(maxyds),numasn(maxyds),cursjd
      INTEGER idurat,ordmod,fidate,npagev,npageh,npromt,numper,numyds,
     1        asntot,sumper,grdtot,topyd ,lowyd ,topidx,lowidx,freptr,
     2        firstp,numasn,ydcalc,mpageh,fiyear,nvruse,asncas,kpad,
     3        cursjd
      LOGICAL lmore ,recalc,prompt,asnsav
C
      COMMON /uasgn/ uasnr,uasnh,uasnc,inasn,outasn,
     1               prnter,autorf,maline
      INTEGER uasnr,uasnh,uasnc,inasn,outasn,maline
      LOGICAL prnter,autorf
C
      COMMON /basgn/ nextp,zshpcl,shpord,sumasn,valasn(maxper),
     1               codasn(maxper)
      INTEGER nextp,zshpcl(6),valasn,codasn,sumasn
      INTEGER*4 shpord
      CHARACTER shpcls*mccls
      EQUIVALENCE (shpcls,zshpcl)
C
```

ALIAS FORTRAN INCLUDE FILES

FILE ASGNDESC

```
C
C  DESCRIPTION OF PARAMETERS IN /asgn/
C     mperh - maximum number of display periods (20)
C     maxyds- maximum number of yards (99)
C     maxper- maximum number periods
C     lcasgn- length of /casgn/ (char)
C     lnasgn- length of /nasgn/ (*2 words)
C     lbasgn- length of /basgn/ (*2)
C     zdel  - nonprinting delimiter character
C     mccls - shpcls character length
C     mccls1- 11, the length of the allowable class input
C     mcyds - ydname character length
C
C  IN /casgn/ ***** character PART OF ASSIGNMENTS WORKING COMMON
C     scname- scenario name (*16)
C     perlbl- period label (*8) [ie. "PERIOD:"]
C     duratn- period duration (*8) [ie. "YEARS"]
C     cpageh- horizontal page character (ie. "A")
C     spromt- short prompt characterss (*2)
C     apromt- prompt sequence (*16 max)
C     Ydname- name of each yard (*12)
C
C  IN /nasgn/ ***** NUMERIC PART OF ASSIGNMENTS WORKING COMMON
C     idurat- period duration indx ([1-6]: [FYr CYr Qtr Month Week Day])
C     ordmod- the first 5 bits (from the right) are currently in use.
C               1: ship class ordering; off=alpha, on=input
C               2: ship yard  ordering; off=alpha, on=input
C               3: use historical section of database
C               4: use current section of database
C               5: use projected section of database
C     fidate- first qtr/wk/day in date row
C     npagev- vertical page number
C     fiyear- starting year
C     asnsav- .T. if last command modified the direct access file
C     npageh- horizontal page number
C     npromt- number of characters in apromt
C     numper- number of periods defined
C     numyds- number of yards defined
C     asntot- total assignments count
C     grdtot- total ships count
C     cursjd- cursor open on the job description file: for ckpf
C     lmore - .T. if more in last yard
C     topyd - loc of top yard displayed
```

```
C     lowyd - loc of lowest yard displayed
C     topidx- top index of top yard
C     lowidx- low index of low yard
C     recalc- .T. if page recalc needed
C     ydcalc- yard index for recalc.
C     freptr- free chain pointer
C     mpageh- max horizontal page number
C     prompt- T:interactive, F:command
C     asncas- bits describing uppercases to be done after input
C             bit 1: uppercase the first char of input yard names
C                 2: uppercase entire input yard names
C                 3: uppercase the first char of input classes
C                 4: uppercase entire input classes
C     Sumper- total ships by period
C     Firstp- assignment chain heads
C     Numasn- total number of assignment rows, by yard
C     nvruse- the lowest unused record
C
C  IN /basgn/ ***** BUFFER PART OF ASSIGNMENTS WORKING COMMON
C     nextp - next buffer in chain
C     shpcls- ship class (*12)
C     sumasn- total assignments in this buffer
C     Valasn- the value of each assignment
C     Codasn- code for each assignment
C
C  IN /uasgn/ ***** UNSTORED PART OF ASSIGNMENTS WORKING COMMON
C     uasnr - unit number for direct access file records
C     uasnh - unit number for help file
C     uasnc - unit for /asgn/ save
C     inasn - input logical unit number
C     outasn- output unit number
C             the first input character is treated as a default
C             and prints out as a blank (right now, New)
C     prnter- .T. if asnclr should act like a line printer
C     autorf- .T. if auto-refresh is on (see asnref)
C     maline- maximum number of buffer lines which fit on screen
C
```

FILE ASHLDR

```
C   include file ashldr
      common /ashldr/hldval(maxper),hldcod(maxper)
      integer hldval,hldcod
C
C        holding buffer for per-period assignments and series codes
c        used by outbound to tranfer to extended memory
C
```

FILE ASJD

```
C! include file asjd
      common /asjd/ndesc,jclas,jyard,jjtyp,jstyp,jcust,jgrp,
     1jcomn,jmethd,jda,jaa,jas,jsk,jkl,jld,jdadd,jdawd,jtunt
      dimension jclas(asmpft),
     1          jyard(asmpft),jjtyp(asmpft),jstyp(asmpft),
     2          jcust(asmpft),jgrp(asmpft),jcomn(asmpft),
     3          jmethd(asmpft),jda(asmpft),jaa(asmpft),jas(asmpft),
     4          jsk(asmpft),jkl(asmpft),jld(asmpft),
     5          jdadd(asmpft),jdawd(asmpft)  ,jtunt(asmpft)
      character jclas*10,jyard*8,jjtyp*6,jstyp*6,jcust*8,jgrp*10
      character*6 jmethd,jtunt
      integer ndesc,jcomn,jda,jaa,jas,jsk,jkl,jld,jdadd
      integer*4 jdawd
      integer jdalin(asmpft,5)
      equivalence (jdalin,jaa(1))
C
C         job descriptions buffer: holds one per class in a
C         complexity group
C
C        jdalin   alternative address for building period intervals
C       ndesc    number of description records currently held
C       jclas    class of job desc
C       jyard    yard it applies to, or ANY
C       jjtyp    job type it applies to
C       jstyp    series type it applies to
C       jcust    customer for this job
C       jgrp     complexity-group of this job
C        jcomn   commissioning number
C        jmethd  construction method
C       jda      design to award time
C       jaa      appropriation to award time
C       jas      award to start
C       jsk      start to keel time
C       jkl      keel to launch time
C       jld      launch to delivery time
C       jdadd    days added to a ship's life by this job
C       jdawd    default award date in a year: for dpsmode=awards
C                and time units = years
C       jtunt    time units schedule intervals given in
```

FILE ASNOCR

```
C! include file asnocr
      common /asnocr/jomain,jodprj,joprj2,joflag,jdesc,
     1                jodol,jolbr,joemp,jomr,jomd,jocom,
     2                jodolf,jolbrf,joempf,jomrf,jomdf,jocomf
      integer jomain,jodprj,joprj2,joflag,jdesc,
     1        jodol,jolbr,joemp,jomr,jomd,jocom
      logical jodolf,jolbrf,joempf,jomrf,jomdf,jocomf
C
C                  CURSOR FOR
C         jomain   ncjodat.projj main file when copy algorithm used
C         jodprj   projected jobs  (ncjodat.projj
C         joprj2   same
C         joflag   same, select by scenario,yard,class,jobtyp
C                        where flag="YES"
C         jdesc    job descriptions (ncjdat.descj
C         jodol    ncjodol.projj
C         jolbr    ncjolbr. "
C         joemp    ncjoemp. "
C         jomr     ncjomr.  "
C         jomd     ncjomd.  "
C         jocom    ncjocom. "
C                  OPEN STATUS FLAGS FOR EIGHT OF ABOVE CURSORS
C         jodolf
C         jolbrf
C         joempf
C         jomrf
C         jomdf
C         jocomf
C
```

FILE ASNVLD

```
C
C  include file /asnvld/
      PARAMETER mvcls=200, mvyds= 99, mcvcls=10, mcvyds= 8,
     1          mcdchr=10, mjtchr=10, mcjt  = 6, mccd  = 6
      COMMON /asnvld/ vldcls(mvcls),nvcls,vldyds(mvyds),nvyds,
     1 jtidef,jtchar,ljtchr,jtname(mjtchr),jtvld(mjtchr),
     2 jttype(mjtchr),cdidef,cdchar,lcdchr,cdname(mcdchr)
      CHARACTER cdchar*mcdchr,jtchar*mjtchr,cdname*mccd,jtname*mcjt
      CHARACTER jttype,vldcls*mcvcls,vldyds*mcvyds
      INTEGER   cdidef,jtidef,nvcls,nvyds,lcdchr,ljtchr
      LOGICAL   jtvld
C  holds lists of valid classes/yards from liston (VALCLS,VALYDS)
C     vldcls- list of valid shipclasses, of length nvcls
C     vldyds- list of valid yards, of length nvyds
C  and holds legal code characters and their translations (.LEGALS):
C     cdchar- character CoDes, like Lead, etc...of length *lcdchr
C     cdname- names corresponding to character codes; length lcochr
C     cdidef- default series code location
C     jtchar- character jobtypes, like New, Repair, etc.; *lcdchr
C     jtname- names corresponding to the job type codes; length lcdchr
C     jtidef- default jobtype code location
C     jtvld-  True if menu system (liston) gives this type as valid
C     jttype- N for job types with data in NC relations, O for REpairs
C
```

FILE ASOPRM

```
C! include file asoprm
      parameter tupunt=21    , asomxc=50 , asomxg=20
      parameter asmshp=200   , asmpft=10 , asmndt=6
      parameter exunit=24,exout=25
      common /asoprm/asucur,asuhis,asamod,asabas,asdbas,
     1               astunt,asfstd,aslstd,memid,perlen
      logical asucur,asuhis
      integer perlen(300)
      integer memid,astunt,asamod,asabas,asdbas
      integer*4 asfstd,aslstd
C
C        general as_igner outbound variables and parameters
C        PARAMETERS
C     asomxc   maximum new-construction classes in a single yard
C     asomxg   maximum classes in a single complexity group in a yard
C     asmshp   maximum ships record buffer can hold at once
C     asmpft   max planning factor tuples for one yard-class-jtyp combo
C     tupunt   unit number for tupfil
C     exunit   unit number for RELATE execution file newhul
C     asmndt   number of schedule dates (award,start,etc) for nc jobs
C        VARIABLES
C     asuhis   true if historical data brought in on inbound
C     asucur   true if current data brought in on inbound
C     perlen   length of each display period in days
C     asamod   schedule adjust basis mode: 1=none,2=class,3=cmplx grp
C     asabas   adjust basis date code:1=award.....5=delivery
C     asdbas   display basis date code:1=award.....5=delivery
C     astunt   time unit code: 1=fyear,2-cyear....6=days
C     asfstd   first day of first bufasn period (clarified ddate)
C    memid    id code for extended memory segment
```

```
C! include file asrbuf
C  NOTE: asoprm must be included above
      common /asrbuf/rnptr,rlptr,rclas,rcode,rdispd,radjd,rfirst,rlast,
     1               rperd
      common /asrcls/rclasf,nscl
      integer rnptr(asmshp),rlptr(asmshp),rclas(asmshp),rcode(asmshp)
      integer rperd(asmshp)
      integer*4 rdispd(asmshp),radjd(asmshp)
      integer*4 rfirst(asmshp),rlast(asmshp)
      integer rclasf(asomxg),nscl(asomxg)
C
C        assigner outbound 1 ship--1 record buffer
C        holds variables used in generating key sched dates
C        for each ship in a complexity-group in a yard
C     rlptr    pointer to last ship in same class
C     rnptr    pointer to next ship in same class
C     rclas    id number of class of ship i; references rclasn
C     rcode    code number of class of ship i; e.g. 'LEAD'
C     rdispd   relate*4 format date that display of ship was based on
C     radjd    date that schedule adjustment of ship is to be based on
C     rfirst   first date that radjd may be set to
C     rlast    last date that radjd may be set to
C     rperd    period ship's displaydate falls in
C    -rclasn-  for character name of each class see hldcls
C     nscl     number of ships in each class
C     rclasf   pointer to first record holding ship of class j
C
```

FILE ASTFR

```
C! include file astfr
C
      parameter asdtst = 31
      common/astfr/fscen,fclas,fhull,fyard,fcom,fjtyp,fjstyp,fcust,
     1             fcmthd,fappr,faward,fstart,fkeel,flaun,fdeliv,fcomm,
     1            fdad,
     2             fflag,forder,fhmap,fddat,fedat,feby
      common/astup/tscen,tclas,thull,tyard,tcom,tjtyp,tjstyp,tcust,
     1             tcmthd,tappr,taward,tstart,tkeel,tlaun,tdeliv,tcomm,
     1             tdad,
     2             tflag,torder,thmap,tddat,tedat,teby
      common/asvar/eoreln
C
      logical eoreln
      character fscen*12,tscen*12,fclas*10,tclas*10,fyard*8,tyard*8
      character fjtyp*6,tjtyp*6,fcmthd*6,tcmthd*6,feby*8,teby*8
      character fcust*8,tcust*8,fjstyp*6,tjstyp*6,fflag*4,tflag*4
      integer fhull,thull,fcom,tcom,fdad,tdad
      integer*4 fappr,tappr,faward,taward,fstart,tstart
      integer*4 fkeel,tkeel,flaun,tlaun,fdeliv,tdeliv,fcomm,tcomm
      integer*4 forder,torder,fddat,tddat,fhmap,thmap,fedat,tedat
C        equivalences
      integer tfrali(60),tupali(60)
      equivalence (fscen,tfrali),(tscen,tupali)
      character*46 fld05(4)
C
C        records for reading/writing of tupfil (tfr) and ncjodat
C        the records are identical; each pair of variables will
C        be described only once
C
C     asdtst   parameter giving location of award in aligned arrays
C     fscen    scenario name
C     fclas    class name
C     fhull    hull number
C     fyard    yard name
C     fcom     commissioning number
C     fjtyp    job type
C     fjstyp   series type (lead,follow..)
C     fcust    customer
C     fcmthd   construction method
C     fappr    appropriation date
C     faward   award date
C     fstart   start date
```

```
42.00 C      fkeel    keel date
43.00 C      flaun    launch date
44.00 C      fdeliv   delivery date
45.00 C      fdad     days added to the life of the ship by this commissioning
46.00 C      fflag    flag setting (YES, NO)
47.00 C      forder   asnorder value
48.00 C      fhmap    the bit map indicating which relations hold hardwire
49.00 C               data for this ship
50.00 C      fddat    data date
50.10 C      fedat    entry date
50.20 C      fcomm    commissioning date
50.30 C      feby     name of user running assigner; entry_by
51.00 C
52.00 C      eoreln   End Of RELatioN.  The end-of-file indicator for
53.00 C               ncjodat.projj must be global.
54.00 C
```

ALIAS FORTRAN INCLUDE FILES

FILE BGPMTR

```
C
C include file bgpmtr---BATTLE GROUP REPORT GENERATOR PARAMETERS
        PARAMETER MXTYPE = 100
        PARAMETER MXCHOICE = 20
        PARAMETER MXFUNC = 50
        PARAMETER MXMKUP = 400
        PARAMETER MXGROUP = 20
C
```

FILE BGTITL

```
C
C include file bgtitl---BATTLE GROUP REPORT GENERATOR TITLE
        CHARACTER SECTITL*20
        COMMON/BGTITL/SECTITL
C       sectitl  current battle group report section title
C
```

ALIAS FORTRAN INCLUDE FILES

FILE CHLST

```
C
C  include file  chlst---MNUG block; list processed choice menus
      CHARACTER CMNLST*LNAME
      INTEGER*4 PMPTR,CMPTR
      INTEGER NCMENU
      COMMON /CHLST/ NCMENU, CMNLST(MXMENU), CMPTR(MXMENU)
     +   ,PMPTR(MXMENU)
C
```

FILE CMENU

```
C
C  include file CMENU---MNUR block; data for current choice menu
      CHARACTER MNTXT*DIMNAME,MTTXT*LLINE
      INTEGER*4 IDOPT, OPTPTR, OPTTXT, MHPTR, OPHPTR
      INTEGER   OPSECI, IDMENU, NOPTIO, OPTTYP, MHLEN, OPHLEN
      COMMON/CMENU/ IDMENU, MHLEN, NOPTIO, OPTTYP(15), OPHLEN(15),
     1   OPTPTR(15), IDOPT(15), MHPTR, OPHPTR(15), OPTTXT(15)
     1    , MTTXT, MNTXT, OPSECI(15)
C
        Id number current menu, number lines menu level help text
        number options on menu, type each option, number lines
        help each option, id for each option (a menu id),
        pointers to option rnproc call goto indexes for process
        options pointer to menu help text,
        pointers to option help text, pointer to option text
        for menu, pointer to menu title text, pointer to menu name,
        security index each option (indexes element of modnum
        in /uzrprv/.
```

FILE COLUSE

```
C
C include file coluse
      common /coluse/ clused(nltrcl,mxltyp)
      logical clused
C
C        This common block contains information about free space
C        in list type relations.
C     clused   Dimensioned (# of columns in each list type relation) by
c              (maximum number of list type relations), this array is
c              a set of flags indicating which columns are in use
C              (.true.).
C
```

ALIAS FORTRAN INCLUDE FILES

```
C
C  include file comcfl
      parameter mxcflv = 10
      INTEGER CFRECS,INCFL, OUTCFL,CFLEVL,CFTREC
      LOGICAL INUSE, BUILDN ,CFECHO
      CHARACTER*8 CFMU1,CFNAM1
      COMMON/COMFLE/ INCFL,OUTCFL,INUSE,BUILDN,CFECHO,CFLEVL
     1               ,CFRECS(mxcflv),CFTREC(mxcflv),CFMU1,CFNAM1
C     mxcflv  maximum nesting level of executing command files
C     incfl   file # from which command file should be read
C     outcfl  file # to which command file should be written
C     inuse     true only if input is being read from incfl
C     buildn    true only if command file is begin built
c     cfecho true if readln should echo input to screen
C     cflevl  current command file execution nesting level
C     cfrecs  count of records read from/written to comfile
C     cftrec  number of records in file execing at level cflevl
C     cfmu1   menu id for command file exec-ing at cflevl=1
C     cfnam1  name of command file exec-ing at cflevl=1
```

FILE CONST

```
C
C  The following are constant values... /CONST/
      PARAMETER
     1   cr     = %15C           , lf     = %12C          ,
     2   ff     = %14C           , large  = %077777        ,
     3   llarge = %17777777777J  , pi     = 3.141592654    ,
     4   root2  = 1.414213562    , eps    = 1.0E-75        ,
     5   xlarge = 1.0E+75        , bell   = %7C            ,
     6   nullc  = %0C            , bs     = %10C           ,
     7   largec = %177C
C
```

FILE CURSRS

```
C
C  include file cursrs
      PARAMETER mcchn=22, mcrs=20, mcpth=12
      COMMON /cursrs/ crschn(mcchn),crs(50,mcrs),crspth(mcrs),
     1                crstyp(mcrs) ,crsidx(mcrs),crsxl(mcrs)
      INTEGER crschn,crs,crstyp,crsidx,crsxl
      CHARACTER crspth*mcpth
C     crschn=cursor chain      crs   =cursor pool
C     crspth=cursor path name  crsxl =index calculate length
C     crstyp=[0..3]: 0=not-in-use, 1=relation, 2=synonym, 3=select
C     crsidx=cursor index pointer: 0=no new index, >0 otherwise
C
```

FILE DEBUFF

```
C
C  include file debuff
C     parameter Data Entry MaXimum Buffer Length, Number Buffers,
C                                 FieLds per relation
      parameter demxbl=160,demxnb=15,demxfl=30
      character*255 bflist(2,demxnb)
      common /debuff/ bfpool(demxbl,demxnb),bufrel(demxnb),
     1                bufpth(demxnb),
     1                grpid(demxnb),gpstat(demxnb),gpista(demxnb),
     2                fldsta(demxfl,demxnb),decurs(demxnb),
     3                bfnfld(demxnb),bfallo(demxnb),bflist
      integer bfpool,grpid,fldsta,bfnfld,decurs
      character*26 bufrel,bufpth
      logical gpstat,gpista,bfallo
C        equivalencing
      character*2 cbpool(160,demxnb)
      real rbpool(80,demxnb)
      equivalence (cbpool,bfpool),(rbpool,bfpool)
C        data transfer buffers for the data entry module
C     bfpool   the buffers
C     bufrel   name of relation buffer holds data for
C     grpid    id number of entry group or set buffer belongs to
C     bfstat   buffer entry status; whether its data is to be sent
C     bflist   field list for each buffer
THIS WAS FOR OBSOLETE DATA UPDATING SYSTEM
```

FILE ENVIRN

```
C
C  include file envirn
      CHARACTER*8 GROUPN,RELGPN,starty,groupc*2
      INTEGER LENGPN, ICCTCL(2), LENRLN ,lengpc
      CHARACTER*4 CCTCLR
      LOGICAL develp
      COMMON/ENVIRN/ GROUPN, RELGPN,LENRLN,lengpn,cctclr,starty,
    1 groupc,lengpc,develp
      EQUIVALENCE (ICCTCL(1),CCTCLR)
C    ---system core status info, mainly for dev/prod versions cap
C     groupn   group name in which files are located
C     lengrp   length of group name in characters
C     regpnp   group in which relate files are located
C     lenrln   length of relgpn in characters
C     cctclr   clear screen control characters for terminal
C     icctcl   integer version of cctclr
C     starty   name of terminal type found or given at startup
C     groupc   character suffix for data base relation group names
c              blank if production version; 'T' if development
C     lengpn   length of groupc contents
C     develp   true if development version is being run
C
```

ALIAS FORTRAN INCLUDE FILES

FILE FIELDS

```
C
C  include file fields
C      holds data statements for all includes of the 'rcrd##' type
      data fld01 /":MENUID,RELATION,COLUMN:"/
C
C
```

FILE FLCLASS

```
C
C include file flclass--list of ship class FLRP is dealing with
       INTEGER NCLASS
       CHARACTER CLIST*CNLEN(MAXCLAS)
       COMMON/FLCLASS/ NCLASS, CLIST
C
```

FILE FLCONCH---parameters defining acceptable FLRP input keywords

```
C
C include file flconch
      PARAMETER LKEY=8
      PARAMETER LETOT=4
      PARAMETER ETOT='ETOT'
      PARAMETER LJOB = 3
      PARAMETER JOB ='JOB'
      PARAMETER LEND = 3
      PARAMETER END='END'
      PARAMETER LEITOT=5
      PARAMETER EITOT='EITOT'
      PARAMETER LTYPE=4
      PARAMETER TYPE = 'TYPE'
      PARAMETER LBTOT=4
      PARAMETER BTOT='BTOT'
      PARAMETER LSTART=5
      PARAMETER START='START'
      PARAMETER LPRGLB=5
      PARAMETER PRGLB='PRGLB'
      PARAMETER COMMA=','
      PARAMETER LPARAN='('
      PARAMETER RPARAN=')'
      PARAMETER LTITLE=5
      PARAMETER TITLE='TITLE'
      PARAMETER BLANK=' '
C
      PARAMETER LSTOP = 4
      PARAMETER STOP = 'STOP'
      PARAMETER CONTINUE = '+'
```

FILE FLCONS

```
C
C include file flcons---data for FLRP access to schedule relations
C historical,current,projected construction relations
        PARAMETER HISCNM='-NCJODAT.HISTJ-'
        PARAMETER CURCNM='-NCJODAT.CURRJ-'
        PARAMETER PROCNM='-NCJODAT.PROJJ-'
        PARAMETER CONSFL='-SCENARIO,CLASS,HULL,COMNUM,APPROP,AWARD,DELIV
     +ERY,COMMISSION,DAYSADDED,DATADATE,ENTRY_DATE-'
        PARAMETER CONSKY=
     1   '-SCENARIO,CLASS,HULL,COMNUM,DATADATE:D,ENTRY_DATE:D-'
        COMMON/FLCONS/hccur1,hccur2,cccur1,cccur2,pccur1,pccur2
        character*12 cfscen,cfclas*10
        integer cfhull,cfadlif,cfnumb, cfalin(26)
        integer hccur1,hccur2,cccur1,cccur2,pccur1,pccur2
        integer*4 cfcomd,cfappd,cfawdd,cfdeld,cfdatd,cfdate
C
        equivalence (cfalin(1),cfscen), (cfalin(7),cfclas)
     +           ,(cfalin(12),cfhull),(cfalin(13),cfnumb)
     +           ,(cfalin(14),cfappd),(cfalin(16),cfawdd)
     +           ,(cfalin(18),cfdeld),(cfalin(20),cfcomd)
     +           ,(cfalin(22),cfadlif)
     +           ,(cfalin(23),cfdatd),(cfalin(25),cfdate)
        character*12 ckscen,ckclas*10
        integer ckhull, cknumb, ckalin(17)
        integer*4 ckdatd,ckdatn
        equivalence (ckalin(1),ckscen), (ckalin(7),ckclas)
     +           ,(ckalin(12),ckhull),(ckalin(13),cknumb)
     +           ,(ckalin(14),ckdatd),(ckalin(16),ckdatn)
C
C       -- data reckrd for FLCONS (Force Level CONStruction) data
C
C       hccur1      relate virtual cursor for hstorical cknstruction
C                   relation, path 1
C       hccur2      relate virtual cursor for historical cknstruction
C                   relation, path 2
C       cccur1      relate virtual cursor for current cknstruction
C                   relation, path 1
C       cccur2      relate virtual cursor for current cknstruction
C                   relation, path 2
C       pccur1      relate virtual cursor for projected cknstruction
C                   relation, path 1
C       pccur2      relate virtual cursor for projected cknstruction
C                   relation, path 2
```

```
43.00 C        c_scenr     a scenrio id (f=fieldvalue,k=keyvalue)
44.00 C        c_clas      a ship class id
45.00 C        c_hull      hull number for the ship class
46.00 C        c_appd      ship's appropriation date
47.00 C        c_awdd      ship's award date
48.00 C        c_deld      ship's delivery date
49.00 C        c_datd      date this info was entered into relate
50.00 C        c_adlif     days added to ship life by this constructio
51.00 C        c_numb      construction number first=1
52.00 C
```

FILE FLD03

```
C include file fld03
C
        data fld03 /"-COMFILNAME,COMFILDESC,LASTUSED,CREATOR,DATCREATED,S
     1TARTMENU,NCOMS-"/
C
C          field list for CFLIST (Command File LIST) relation
```

FILE FLD05

```
C! include file fld05
        data fld05 /"+SCENARIO,CLASS,HULL,YARD,COMNUM,JOBTYP,JSTYP,",
     1            "CUSTOMER,CMETHD,APPROP,AWARD,START,KEEL,LAUNCH",
     2            ",DELIVERY,COMMISSION,DAYSADDED,AUTOMOD,ASNORDE",
     3            "R,SUBRELUMAP,DATADATE,ENTRY_DATE,ENTRY_BY+"/
C
C       field list for ncjodat.projj
C
```

FILE FLD06

```
C! include file fld06
        data fld06/"+SCENARIO,CLASS,YARD,JOBTYP,JSTYP,CUSTOMER,",
     1            "COMPLEXGRP,COMNUM,CMETHD,DSGN_AWD,APPROP_AW",
     2            "D,AWD_ST,ST_KL,KL_LN,LN_DL,DAYSADDED,DEFLTA",
     3            "WDAY,TIMUNT+"/
C
C       field list for ncjdat.descj relation
C
```

FILE FLD07

```
*   .   C
*   .   C      ---include file fld07
*   .          data fld07/"+USERNAME,RUNGROUP,USERLEVL,READB,ALTDB,",
*   .         1          "M1,M2,M3,M4,M5,M6,M7,M8,M9,M10,M11,M12,M",
*   .         1          "13,M14,M15,M16,M17,M18,M19,M20,M21,M22,M",
*   .         1          "23,M24,M25,M26,M27,M28,M29,M30,M31,M32,M",
*   .         1          "33,M34,M35,M36,M37,M38,M39,M40,M41,M42,M",
*   .         1          "43,M44,M45,M46,M47,M48,M49,M50+"/
*   .   C
*   .   C      ---field list for DB read into /uzrprv/
*   .   C
```

FILE FLD08

```
*   .   C   include file fld08
*   .          data fld08/"+SCENARIO,CREATOR,RDALLOW,WRALLOW,CREATED,LASTUSED+"/
*   .   C         field list for scenlst relation
*   .   C
```

FILE FLDECM

```
C
C include file fldecm
C contains decommisioning data for all ships, first=1
        PARAMETER DECMNM='-DEACT.MISCJ-'
        PARAMETER DECMFL=
     +  '-SCENARIO,CLASS,HULL,COMNUM,DEACT,DATADATE-'
        PARAMETER DECMKY='-SCENARIO,CLASS,HULL,COMNUM-'
        COMMON /FLDECM/ dccurs
        character*12 dfscen,dfclas*10
        integer dfhull, dfalin(17), dccurs, dfdean
        integer*4 dfdead,dfdatd
        equivalence (dfalin(1 ),dfscen),( dfalin(7), dfclas)
        equivalence (dfalin(12),dfhull),( dfalin(13), dfdean)
        equivalence (dfalin(14),dfdead),( dfalin(16), dfdatd)
C
        character*12 dkscen,dkclas*10
        integer dkhull, dkalin(15), dkdean
        integer*4 dkdatd
        equivalence (dkalin(1 ),dkscen),( dkalin(7), dkclas)
        equivalence (dkalin(12),dkhull),( dkalin(13), dkdean)
        equivalence (dkalin(14),dkdatd)
C
C       -- data record for DECOMM  (DECOMMisioning) data
C
C       d_curs      relate virtual cursor
C       d_scenr     a scenrio id
C       d_clas      a ship class id
C       d_hull      hull number for the ship class
C       d_dean      the number of this ship's deactivation
C       d_dead      the date of this deactivation
C       d_datd      date this info was entered into relate
C
C
```

FILE FLHEAD

```
C
C include file flhead   FLRP page header info
      CHARACTER TITLES*LLONG(MXTITL),PERHED*LLONG(LINPH)
      CHARACTER LABPRG*LENRLB(MAXPRG), UNTOTL*LLONG
      INTEGER NTITLE
      COMMON/FLHEAD/ NTITLE,TITLES,PERHED,LABPRG,UNTOTL
C     -- ntitle  number of lines of title
C     -- title   title text lines
C     -- perhed  period header text lines
C     -- labprg  label for program i
C     -- untotl  equal marks above total line
C
```

FILE FLIOC

```
C
C include file flioc     io unit numbers for FLRP
     INTEGER OCNTRL, IOUTFL, FLRPTF
     COMMON/FLIOC/ OCNTRL, IOUTFL, FLRPTF
C    -- ocntrl    force level report input file
C    -- ioutfl    write report here
C    -- flrptf    write permanent report here
C
```

FILE FLJLST

```
C
C include file fljlst
      LOGICAL NONEON
      INTEGER NINCLUD
      CHARACTER*12 INCLST
      COMMON /FLJLST/ NONEON, NINCLUD, INCLST(MAXLST)
C     -- noneon  true if nothing from joblist was on
C     -- ninclud number of joblist condidates on
C     -- inclst  list of joblist on candidates
C
```

FILE FLPAGE

```
C
C include file flpage
       INTEGER INBUF,NONPAG,flpds
C      CHARACTER PAGEBUF*LLONG(LENBUF)
       character pagebuf*llong
       integer ipageb(70)
       equivalence (pagebuf,ipageb)
       COMMON /FLPAGE/flpds,INBUF,NONPAG
C      -- nonpage  number already printed to output page
C      -- inbuf    number already in output buffer
C      -- pagebuf  page buffer, will go to output page
C
C      NOTE: the array version of pagebuf was made scalar
C      and local due to HP memory -strictions; storage is
C      now in exteneded memory segment 8001.
```

ALIAS FORTRAN INCLUDE FILES

FILE FLPERD

```
C
C include file flperd
         INTEGER NPERIOD
         INTEGER*4 DATPER
         COMMON/FLPERD/ NPERIOD,DATPER(MXPERD)
C        -- datper   any ship active on datper(i)is in period(ı
C        -- nperiod  number of periods being examined
C
```

FILE FLPMTR

```
C
C include file flpmtr           FLRP parameters
      PARAMETER CNLEN=12
      PARAMETER MAXCLAS=100
      PARAMETER MXTITL=10
      PARAMETER LLONG=132
      PARAMETER BLTITL='
     +
     +                  '
      PARAMETER LENLLB=12
      PARAMETER LENRLB=12
      PARAMETER WDTCOL=5
      PARAMETER LINPH=3
      PARAMETER MAXLST=100
      PARAMETER LENBUF=80
      PARAMETER LENPAG=60
      PARAMETER MXPERD=20
      PARAMETER MAXPRG = 4
      PARAMETER MXTOTL=15
C
```

```
C
C include file flrjob
C contains repair job data for historical, current, projected ships
        PARAMETER HISJNM='-REJODAT.HISTJ-'
        PARAMETER CURJNM='-REJODAT.CURRJ-'
        PARAMETER PROJNM='-REJODAT.PROJJ-'
        PARAMETER RJOBFL=
     +  '-SCENARIO,REJOBT,CLASS,HULL,JOBID,START,DELIVERY,DAYSADDED,DATA
     +DATE-'
        PARAMETER RJOBK1='-SCENARIO,REJOBT,CLASS,JOBID,
     +  DATADATE:D,ENTRY_DATE:D-'
        PARAMETER RJOBK2=
     1  '-SCENARIO,CLASS,HULL,JOBID,DATADATE:D,ENTRY_DATE:D-'
        COMMON /FLRJOB/ hjcur1,hjcur2,cjcur1,cjcur2,pjcur1,
     +                  pjcur2
        integer hjcur1,hjcur2,cjcur1,cjcur2,pjcur1,
     +                  pjcur2
        character jfscen*12,jfclas*10,jfjobn*6
        integer jfhull, jfalin(26), jfadda ,jfjobid
        integer*4 jfbegd,jfendd,jfdatd
        equivalence (jfalin(1),jfscen), (jfalin(7),jfjobn)
        equivalence (jfalin(10),jfclas), (jfalin(15),jfhull)
        equivalence (jfalin(16),jfjobid)
        equivalence (jfalin(17),jfbegd), (jfalin(19),jfendd)
        equivalence (jfalin(21),jfadda), (jfalin(22),jfdatd)
C
        character jk1scen*12,jk1clas*10,jk1jobn*6
        integerjk1alin(22),jk1jid
        integer*4 jk1datd,jk1date
        equivalence (jk1alin(1),jk1scen), (jk1alin(7),jk1jobn)
        equivalence (jk1alin(10),jk1clas)
        equivalence (jk1alin(15),jk1jid), (jk1alin(16),jk1datd)
        equivalence (jk1alin(18),jk1date)
C
        character*12 jk2scen,jk2clas*10
        integer jk2hull, jk2alin(20), jk2jid
        integer*4 jk2datd,jk2datn
        equivalence (jk2alin(1),jk2scen), (jk2alin(7),jk2clas)
        equivalence (jk2alin(12),jk2hull), (jk2alin(13),jk2jid)
        equivalence (jk2alin(14),jk2datd),(jk2alin(16),jk2datn)
C
C       -- data record for FLRJOB( Repair JOBs) data
C
```

```
43.00 C          j_curs      relate virtual cursor(f=fieldvalue,k1=key1
44.00 C          j_scenr     a scenrio id           ,k2=key2)
45.00 C          j_clas      a ship class id
46.00 C          j_hull      hull number for the ship class
47.00 C          j_appd      ship's appropriation date
48.00 C          j_awdd      ship's award date
49.00 C          j_deld      ship's delivery date
50.00 C          j_datd      date this info was entered into relate
51.00 C
```

FILE FLTABLS

```
C
C include file fltabls
        INTEGER*4 PRGBEG
        INTEGER NPROGS, FLTABL(MAXCLAS,MXPERD,MAXPRG)
        COMMON /FLTABLS/ NPROGS,FLTABL,PRGBEG(MAXPRG)
C       -- nprogs   number of program types studied
C       -- fltabl   force level table, number of that class in
C                   that period for that program type
C       -- prgbeg   date of program start
C
```

FILE FLTOTL

```
C
C include file fltotl
          CHARACTER TOTID*LENRLB
          INTEGER NTOTAR,TOTALS,INTOTL
          COMMON/FLTOTL/NTOTAR, TOTALS(MXTOTL,MXPERD)
     +               ,TOTID(MXTOTL),INTOTL(MXTOTL)
C         ntotal   number of rows(arrays) in total being computed
C         totals   holds all totals for each period in computation
C         totid    holds label for total
C
```

FILE FLVALU

```
C
C include file flvalu
      LOGICAL FLKEEP
      INTEGER*4 FLBXER
      INTEGER*4 FLFXER
      CHARACTER*   8 FLMRET
      CHARACTER*   8 FLPLEN
      CHARACTER*   8 FLINFR
      CHARACTER*   8 FLPMLS
      CHARACTER*  24 FLRJOB
      COMMON/FLVALU/FLKEEP,FLBXER,FLFXER,FLMRET,FLPLEN,FLINFR
     +       ,FLPMLS, FLRJOB
C
```

FILE FUNCBG

```
C
C include file funcbg
      INTEGER NFUNC,FDEFINE(MXCHOICE,MXFUNC)
      CHARACTER FNMLST*CNLEN(MXFUNC)
      COMMON/FUNCBG/NFUNC,FDEFINE,FNMLST
C     nfunc  number of funtional families defined
C     fdefine for each function,this holds index into type arrays
C             of type which will perform the function in order of
C             choice, 1=highest priority
C     fnmlst  function names for cross referencing into fdefine
C
```

```
C**include file gntupd
      parameter ntyp = 2 , mxtlen = 28, nfil = 6, mxtlenb = 56
      parameter mxvc=200,mxvy=100,mxvj=20
      common /gntupd/nrels,nomore(nfil),itup(mxtlen,nfil),gnfld(ntyp),
     1               klenb,gncurs(nfil),gnfile(nfil),gntype(nfil),
     2               vlclas(mxvc),vlyard(mxvy),vljobt(mxvj),
     3               nowcls(nfil),nowyrd(nfil),nowjob(nfil),
     4               nvclas      ,nvyard      ,nvjobs
      integer nrels,itup,klenb,gncurs
      character gnfld*62
      character gnfile*18,ctup*mxtlenb(nfil)
      character vlclas*10,vlyard*8,vljobt*6
      integer nowcls,nowyrd,nowjob
      integer nvclas,nvyard,nvjobs
      logical nomore
      equivalence (itup,ctup)
C
C     ---static storage for gntup routine
C     nrels    number of relations to be accessed (max nfil)
C     nomore   true if no more data in relation i
C     itup     storage for current tuples
C     klenb    length of key section of tuple in bytes (for chash)
C     gncurs   cursor indexes for each relation
C     gnfile   name of each relation
C     gnfld    field list for two relation types
C     gntype   type of each relation
C     vl:clas,yard,jobt  lists of valid field values
C     nv:clas,yard,jobs  number of members on each valid list
C**ENDBLK
C
```

FILE GROUPBG

```
C
C include file groupbg
      INTEGER NGROUP,GRPFILP(MXPERD,MXGROUP),GMKPTR(MXGROUP)
      INTEGER DGRPLEVL(MXPERD,MXGROUP),AGRPLEVL(MXPERD,MXGROUP)
      CHARACTER GRPLAB*LENRLB(MXGROUP),GRPLST*CNLEN(MXGROUP)
      COMMON/GROUPBG/NGROUP,GRPFILP,GMKPTR,DGRPLEVL,AGRPLEVL,
     +       GRPLAB,GRPLST
C     ngroup    number of battle groups
C     grpfilp   group's fill priority order(1high)
C     dgrplevl  desired number of this group
C     agrplevl  actual number of this group found
C     grplab    label for output group lkine
C     grplst    list of group names
C
```

FILE INCPAR

```
C
C include file incpar        MNUR parameters
      PARAMETER MXMENU = 100
      PARAMETER MAXOPT = 15
      PARAMETER LLINE = 72
      PARAMETER LTEXT = 70
      PARAMETER SNAME = 6
      PARAMETER LNAME = 6
      PARAMETER DIMNAME = 8
      PARAMETER LHTXT = 800
C     mxmenu    maximum number of choice or parameter menus
C     maxopt     maximun options per choice or param. menu
C     lline      length of an input line
C     ltext      length of input descriptive text
C     lname      max. length of a parameter name
C     sname      max. length of a subroutine name
C     dimname    dimension of a name>=lname; divisible by 12
C     lhtxt      max. length of a menu's help text
C
```

ALIAS FORTRAN INCLUDE FILES

FILE INDEXS

```
C
C  include file indexs          DBIF index tracking info
      PARAMETER michn= 6,midx= 4,mcidx=132
      COMMON /indexs/ idxchn(michn),idx(midx)
      INTEGER idxchn
      CHARACTER idx*mcidx
C     idxchn=index chain    idx=index pool
C
```

FILE INPUTL

```
C
C include file inputl            CORE (READLN) LAST LINE BUFFER
       INTEGER LENLN
       CHARACTER LASTLN*LLINE
       COMMON / INPUTL/ LENLN,LASTLN
C      lenln   non-blankes length of lastln
C      lastln  last line read from command input file
C
```

FILE IO

```
C
C  include file io              TERMINAL I/O UNIT NUMBERS
      COMMON /io/ in,iout,itty
      INTEGER in,iout,itty
C
C     in    =standard INput unit       iout  =standard OUTput unit
C     itty  =standard user input unit
C
```

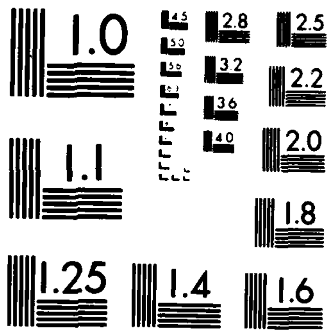
FILE IOC

```
C
C  include file ioc:
      parameter salp = 8,  daisy= 7, termnl= 6
      parameter syshlp=49, sysmp=48, dpdesc=51, drunp=52
      parameter dcmen=53,  dmroot=54, dpxref=55, dpequi=56
      parameter dpdec=57,  dpmen=58,  dhtxt=59,  ddtxt=60
      parameter deunit=22, idunit=23
C     units 24-27 used by scenario system
      parameter modhlp=28
       INTEGER IN,IOUT,IOUTLP,ITTYIN,ITTYOU,IOCEXTRA(10)
       COMMON/IOC/ IN,IOUT,IOUTLP,ITTYIN,ITTYOU,IOCEXTRA
C      salp    unit number for PMS 392 line printer
C      daisy   unit number for SEA 90 daisy wheel printer
C      termnl  unit number for $STDLIST
C      ioutlp  file assigned to desired output printer
C      in      file from which input is expected
C      iout    file to which output is written
C      drunp   file to which subroutine runprocs is written
C      ittyin  file assigned to input from screen file
C      ittyou  file assigned to output to screen file
C      dpdecs  file to which common/pdecs/ is written
C      dlistm  file to which list memory is written
C      dmroot  file to which common/mroot/ is written
C      dpxref  file to which parameter menu field lists
C              and  pointers are written
C      dpequi file to which equivalance statement between
C              pvalue and parameter names is written
```

ALIAS FORTRAN INCLUDE FILES

```
C       dpdec    file to which parameter type statements
                 are written
C       deunit   data maintenance subsystem def file unit
C       idunit   unit on which unique id code generator file opens
C        iocextra   use this space when adding to common block
                       rather than recompile the world.
C       modhlp   unit module help/menu text file is opened on
C
```

AD-A150 423   ALIAS (ACQUISITION AND LOGISTICS INFORMATION AND   4/7
ANALYSIS SYSTEM) MAINTEN..(U) DECISION-SCIENCE
APPLICATIONS INC ARLINGTON VA M S CAREY ET AL.
UNCLASSIFIED   31 OCT 84 DSA-593-VOL-2 N00014-82-C-0813   F/G 15/5   NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

FILE LINKM

```
C
C  include file linkm   ---list memory global storage
      PARAMETER lenlkm =  4000, llinkm =  8006
      COMMON  /linkm/ iavail, iavind, nbcell, linkm(lenlkm)
      INTEGER*4       iavail, iavind, nbcell, linkm
C  iavail- free cell chain ptr  iavind- 'available' indicator value
C  nbcell- # words in list mem  linkm - Linked list memory array
C
```

FILE LISTYP

```
C
C include file listyp
      common /listyp/ nltyps, ltyps(mxltyp)
      integer nltyps
      character*8 ltyps
C
C        Common listyp contains a list of all valid list menu types.
C     nltyps   Number of List menu TYPeS generated by mnugen.
C     ltyps    names of List menu TYPes generated by mnugen.
C
```

ALIAS FORTRAN INCLUDE FILES

FILE LMENU

```
C
C include file lmenu            MNUR---pointer for current list menu
      CHARACTER*8 LMID, LMRELT
      INTEGER*4  ITXTP, LHPTR, LHLEN
      COMMON /LMENU/ ITXTP,LMRELT, LMID, LHPTR, LHLEN
C     itxtp   pointer to list menu's title text
C     lmid    unique list menu id for data base
C     lmrelt  relation type of list menu for current parameter
C     lhptr   pointer to list help in dhtxt
C     lhlen   number of lines of help text
C
```

FILE LPRNTS

```
C
C  include file lprnts          DIAGNOSTIC SWITCHES
      COMMON /lprnts/ ioutp,lprnts(160)
      INTEGER ioutp
      LOGICAL lprnts, lprnt
C
```

FILE LTYPAC

```
C
C include file ltypac
      common /ltypac/ itypcr,tflist
      integer itypcr
      character*30 tflist
C
C         Common List TyPe relation ACcess allows a single open and
c         field-list construction to occur when a given list menu
c         is to be updated.  Saving the field-list, which contains
c         the field name of the proper status column, saves much
c         redundant DB accessing.
c     itypcr   cursor number for the current list type relation
c     tflist   field list for the current list type menu
C
```

FILE LVAL

```
C
C   include file lval:            MNUR---list menu candidates/statuses
      PARAMETER MAXLST = 200
      INTEGER NLITMS, LENHLP
      INTEGER*4 LHLPPTR
      CHARACTER*12 CNAME
      LOGICAL STATUS
      COMMON/LVAL/NLITMS, CNAME(MAXLST), STATUS(MAXLST)
     +   ,LHLPPTR, LENHLP
C
```

ALIAS FORTRAN INCLUDE FILES

FILE MKUPBG

```
C
C include file mkupbg                      BGRP---aggregation control data
        INTEGER LASTREC,BGMAKUP(MXMKUP,3)
        COMMON/MKUPBG/LASTREC,BGMAKUP
C       lastrec  last row of bgamkup filled
C       bgmakup  holds group makeup definitions, 1 col=ptr
C                to next row in bgmakup holding next part of
C                definition, 0 if last. 2 col index into funtion
C                arrays, 3 col # of this function needed to makeup
C                one battlegroup
C
```

FILE MNUPRM

```
C
C include file mnuprm
       parameter mxlfld=12, mxltyp=50, mxrels=200, nltrcl=09
C      parameter maxmlt=nltrcl/mxscen; # columns in list type rel/mxscen
       parameter mxlmlt=5
C
C         Contains parameters used exclusively by the menu system.
C      mxlfld   MaXimum List menu relation candidate FieLD size.
C               Be sure to alter hardwired creation call in glstrl if
C               this parameter's value is changed.
C      mxltyp   MaXimum number of List menu TYPe relations.
C      mxrels   MaXimum number of RELationS creatable by mnugen.
C      nltrcl   Number of List Type Relation CoLumns.
C      mxlmlt   MaXimum List Menus per List Type
C
```

FILE MROOT

```
C
```

```
C  include file mroot:          MNUR---pointer to root menu
       INTEGER*4 MRTPTR
       COMMON /MROOT/MRTPTR
C
```

ALIAS FORTRAN INCLUDE FILES

FILE PARAMS

```
C
C include file params
        parameter rnmxch=6, mxscen=10
C
C               Contains system-level parameters.
C       rnmxch   Relation Name MaXimum Characters.  May be no more
C                than 7 for HP RELATE.
C       mxscen   MaXimum number of SCENarios.
C
```

FILE PARLST

```
C
C  include file parlst:          MNUG---list of parameter menus read
       CHARACTER PMNLST*LNAME
       COMMON /PARLST/ PMNLST(MXMENU)
C
```

FILE PCREAT

```
C
C  include file pcreat:         MNUG---data for parameter storage setup
      INTEGER ISTRC, NPADS, IDFLTV
      PARAMETER MXLSTR = 300
      CHARACTER*1 STRCTR, CDEFLT*4(100)
      REAL DEFLTV
      COMMON / PCREAT/ ISTRC, DEFLTV(100), STRCTR(MXLSTR)
     1  ,NPADS,IDFLTV
      EQUIVALENCE (CDEFLT(1),DEFLTV(1) )
C     instrc    # of characters in strctr
C     npads     # of padding variables used
C     idlftv    # of words used in defltv
C     defltv   default values for relation  command in listqeueu
C     cdeflt   character rep. of defltv
C     listqu  actual queue of list commands,
C         first out = listqu(inque),
C
```

FILE PDESC

```
C
C   include file pdesc :
      INTEGER*4 MPTXTP, PHPTR
      INTEGER   MPINDX, MPTYPE, MPLEN, PHLEN
      COMMON/PDESC/ MPTXTP(MAXOPT),MPLEN(MAXOPT),PHLEN(MAXOPT),
     1  MPINDX(MAXOPT), MPTYPE(MAXOPT), PHPTR(MAXOPT)
C
```

FILE PGSYS

```
C  include file pgsys
      parameter pgunit=29
      common /pgsys/pgatln,pgnext,pgout,pgllen,pgplen,pgwfc,
     1              pgomod,pgfmod,pgquit,pgqchr,pglast,pgtop
      integer pgatln,pgomod,pgfmod,pgout,pgllen,pgplen
      integer pglast,pgtop,pgnext
      logical pgquit,pgwfc
      character*1 pgqchr
C        control data for page printer
C     pgatln   location of last line added to page buffer
C     pgnext   location of next free line in buffer
C     pgout    unit to send output to
C     pgllen   line length of output page
C     pgplen   number of lines per output page
C     pgwfc    true if first call to pgwrit has been made
C     pgomod   operating mode:
C              1=prompt user for page feed, print until eopage;prompt
C              2=don't prompt user, print continuously, header each pg
C              3=don't prompt, print continously with header top only
C              4=don't prompt user, let user print on eopage
C     pgfmod   line feed mode: 1 for sclear, 2 for 1h1
C     pgquit   quit? prompting in effect
C     prqchr   recognition character indicating quit
C     pglast   location in buffer of last line guaranteed to fit
C              on this page
C     pgtop    location in buffer of first line awaiting printing
```

FILE PMENU

```
C
C  include file pmenu:          MNUR---current parameter menu info
      CHARACTER RNTXT*DIMNAME,MPTTXT*LLINE
      INTEGER*4  MPHPTR
      INTEGER   IDPMEN, MAXPGI,MPHLEN
      COMMON/PMENU/ IDPMEN,MAXPGI,MPHLEN,MPHPTR
     1                 ,RNTXT, MPTTXT
C
```

FILE PPINDX

```
C
C include file ppindx:          MNUG---parameter set up data
      INTEGER NXTPGI, PVINDX
      COMMON/PPINDX/ NXTPGI, PVINDX
C     nxtpgi  the next parameter to be defined will use
C             index=nxtpgi in the arrays of common/pdesc
C     pvindx  the value of the next parameter to be
C             defined will begin at pvalue(pvindx)
C
```

ALIAS FORTRAN INCLUDE FILES

FILE PRMCRS

```
C INCLUDE FILE PRMCRS            CORE---permanently open cursors
       common /prmcrs/cflcrs,lcrcrs,snucrs
       integer cflcrs,lcrcrs,snucrs
c
C           cflcrs   cursor for command file CFLIST relation
C           lcrcrs   cursor for lccref--list menu crossref
C           snucrs   cursor for snusers--scenarios now in use
```

FILE PVALUE

```
C
C  include file pvalue:          MNUR---parameter values storage
      PARAMETER PTLEN=500
      REAL PVALUE(PTLEN)
      INTEGER IVALUE(2,PTLEN)
      INTEGER*4 DVALUE(PTLEN)
      CHARACTER CVALUE*4(PTLEN)
      LOGICAL LVALUE(2,PTLEN)
      COMMON /PVAL/ PVALUE
      EQUIVALENCE (PVALUE,IVALUE,DVALUE,CVALUE,LVALUE)
C
```

FILE PVDECL

```
                                  CORE---mnug-written decls for /pvalue/ equivs
*    .          CHARACTER*   8 TTYTYP
*    .          CHARACTER*  12 LPUNIT
*    .          LOGICAL PAD1     , PSMNUH
*    .          CHARACTER*   8 PDURAT
*    .          INTEGER*4 DPFRST
*    .          INTEGER*4 DPLAST
*    .          CHARACTER*  24 ASNYDS
*    .          CHARACTER*  24 ASNCLS
*    .          CHARACTER*  24 ASNJTP
*    .          CHARACTER*   8 DISBAS
*    .          CHARACTER*   8 ADJBAS
*    .          CHARACTER*  12 ADJMOD
*    .          CHARACTER*  12 JEPOCH
*    .          CHARACTER*  12 SRTCLS
*    .          CHARACTER*  12 SRTYRD
*    .          CHARACTER*   4 REFRSH
*    .          LOGICAL PAD2     , RPKEEP
*    .          INTEGER*4 RPBXER
*    .          INTEGER*4 RPFXER
*    .          CHARACTER*   8 RPMRET
*    .          CHARACTER*   8 RPPLEN
*    .          CHARACTER*   8 RPINFR
*    .          CHARACTER*   8 RPPMLS
*    .          CHARACTER*  24 RPRJOB
```

FILE PVEQIV

```
                                    CORE---mnug-written equivs to /pvalue/
*   .        EQUIVALENCE   (PVALUE(       1),TTYTYP)
*   .        EQUIVALENCE   (PVALUE(       3),LPUNIT)
*   .        EQUIVALENCE   (IVALUE(1,       6),PAD1     )
*   .        EQUIVALENCE (IVALUE(2,       6),PSMNUH)
*   .        EQUIVALENCE   (PVALUE(       7),PDURAT)
*   .        EQUIVALENCE   (PVALUE(       9),DPFRST)
*   .        EQUIVALENCE   (PVALUE(      10),DPLAST)
*   .        EQUIVALENCE   (PVALUE(      11),ASNYDS)
*   .        EQUIVALENCE   (PVALUE(      17),ASNCLS)
*   .        EQUIVALENCE   (PVALUE(      23),ASNJTP)
*   .        EQUIVALENCE   (PVALUE(      29),DISBAS)
*   .        EQUIVALENCE   (PVALUE(      31),ADJBAS)
*   .        EQUIVALENCE   (PVALUE(      33),ADJMOD)
*   .        EQUIVALENCE   (PVALUE(      36),JEPOCH)
*   .        EQUIVALENCE   (PVALUE(      39),SRTCLS)
*   .        EQUIVALENCE   (PVALUE(      42),SRTYRD)
*   .        EQUIVALENCE   (PVALUE(      45),REFRSH)
*   .        EQUIVALENCE   (IVALUE(1,      46),PAD2     )
*   .        EQUIVALENCE (IVALUE(2,      46),RPKEEP)
*   .        EQUIVALENCE   (PVALUE(      47),RPBXER)
*   .        EQUIVALENCE   (PVALUE(      48),RPFXER)
*   .        EQUIVALENCE   (PVALUE(      49),RPMRET)
*   .        EQUIVALENCE   (PVALUE(      51),RPPLEN)
*   .        EQUIVALENCE   (PVALUE(      53),RPINFR)
*   .        EQUIVALENCE   (PVALUE(      55),RPPMLS)
*   .        EQUIVALENCE   (PVALUE(      57),RPRJOB)
```

FILE PXREF

```
C
C  include file pxref            MNUR---parameter storage retrieval
      PARAMETER DFLEN = 108
C     PARAMETER DFLEN = MAXOPT *(LNAME+1) +2
      CHARACTER PDFLDL*DFLEN
      INTEGER NPRMEN,NXTFLD
      COMMON /PXREF/ PDFLDL,NPRMEN,NXTFLD
C     dflen      delemited field list's max. length
C     pdfldl     delimeted field list for idpmen = i
C     nprmen  number of parameter menus defined
C     nxtfld       next character in pdfldl goes here
C
```

```
C
C  include file queue
      INTEGER INQUE
      CHARACTER*12 LISTQU
      COMMON /QUEUE/ LISTQU(36), INQUE
C     inque    number of list command in listqeueu
C     listqu  actual queue of list commands,
C         first out = listqu(inque),
C         last out = listqu(1)
C
```

ALIAS FORTRAN INCLUDE FILES

FILE RCRD01

```
C
C include file rcrd01
      common /rcrd01/menu01,rltn01,colm01
      character colm01*4
      character*8 menu01,rltn01
      character*34 fld01
      integer alin01(10)
      equivalence (alin01,menu01)
C
C        Common /rcrd01/ provides a record for use in passing
C        data to the lcrref relation.
C     scen01   a scenario id
C     colm01   column number given menu is stored in in relation
C     menu01   a list menu identifier
C     rltn01   a list type relation name
C     fld01    field list for the lcrref relation
C
```

FILE RCRD03

```
C
C include file rcrd03
C
      common /rcrd03/cflnam,cfldsc,cflusd,cflctr,cflctd,cflsmu
     1              ,cflrec
      integer cflrec
      character*8 cflnam,cflctr,cflsmu
      character*42 cfldsc
      character*70 fld03
      real cflctd,cflusd
      integer alin03(38)
      equivalence (alin03,cflnam)
C
C        data record for CFLIST (Command File LIST) relation
C
C        cflnam   command file name
C        cfldsc   command file description
C        cflusd   command file last-used date
```

ALIAS FORTRAN INCLUDE FILES

```
743.00 C        cflctr   command file creator
744.00 C        cflctd   command file creation date
745.00 C        cflsmu   command file start-execution menu
746.00 C        cflrec   number of records in this command file
```

FILE RCRD06

```
C! include file rcrd06
      common /rcrd06/scen06,clas06,yard06,jtyp06,
     1                styp06,cust06,grp06,com06,mthd06,
     2                da06,aa06,as06,sk06,kl06,ld06,dadd06,dawd06,tunt06
      character clas06*10,yard06*8,jtyp06*6,styp06*6,cust06*8,grp06*10
      character tunt06*6,scen06*12,mthd06*6
      integer*4 dawd06
      integer da06,aa06,as06,sk06,kl06,ld06,dadd06,com06
      integer alin06(46)
      equivalence (scen06,alin06)
      character*43 fld06(4)
C
C        holds a tuple returned from the job description relation
C
C     variables are, in order, scenario,class,yard,jobtype,
C      job series type,customer,complexity group,commissioning number,
C      construction method,design-award time,
C      approp-award,award-start,start-keel,keel-launch,launch-delivery,
C      days added to life of ship, default award day in year,
C      and time units planning factors are in
C
```

FILE RCRD08

```
*   .   C  include file rcrd08
*   .         parameter len08 = 22
*   .         common /rcrd08/senam,creatr,racces,wacces,creatd,lstusd
*   .         character senam*12,creatr*8,racces*8,wacces*8
*   .         integer*4 creatd,lstusd
*   .         character fld08*70
*   .         integer alin08(len08)
*   .         equivalence (alin08,senam)
*   .   C
*   .   C        record for communication with scenlst relation
*   .   C     senam    name of scenario
*   .   C     creatr   creator of scenario
*   .   C     racces   read permission ("PUBLIC" or creator name)
*   .   C     wacces   write permission ("PUBLIC" or creator)
*   .   C     creatd   date created
*   .   C     lstusd   date last used
*   .   C
```

ALIAS FORTRAN INCLUDE FILES

FILE READC

```
C
C include file readc    stores last line read by rdln for lwarn
      INTEGER ILINE
      COMMON/READC/ ILINE
C     number of line last read from input file
C     ( for preproc )
C
```

FILE RELNAM

```
C
C include file relnam
      common /relnam/ nrels, rlnams(mxrels)
      character rlnams*8
      integer nrels
C
C        Common RELation NAMes holds the names of all relations
C        to be created by the menu generation processor.
C     nrels    Number of RELationS.  Count of relations created by
C              mnugen.
C     rlnams   ReLation NAMeS.  List of names of relation created by
C              mnugen.
C
```

ALIAS FORTRAN INCLUDE FILES

FILE RPSUBS

```
C
C  include file rpsubs:         MNUG---rnproc creation data
           PARAMETER MAXSUBS= 100
           INTEGER NPROCS
           CHARACTER SUBLST*SNAME
           COMMON/RPSUBS/SUBLST(MAXSUBS), NPROCS
C          nprocs  number of special purpose processes that
C                    have been referenced.
C          sublst   list of all special purpose processes'
C                    subroutine names.
C
```

FILE SCENAR

```
*   .   C  include file scenar          SCENARIO SYSTEM DATA
*   .         parameter sncurs=20
*   .         common /scenar/actsen,cursen(sncurs),dlmsen(sncurs),
*   .        1                wrtprv(sncurs),snwovr
*   .         character*12 actsen,cursen,dlmsen*14
*   .         logical wrtprv,snwovr
*   .         integer alinsen
*   .         equivalence (alinsen,actsen)
*   .   C
*   .   C           current scenario data for application routines
*   .   C     sncurs   max number of relate cursors
*   .   C     actsen   name of current scenario
*   .   C     cursen   scenario key value for relation i
*   .   C     delsen   delimited version of cursen
*   .   C     wrtprv   true if user may write on the current cursor
*   .   C     snwovr   scenario write privelege override; allows
*   .   C              write on cursors regardless of scenario
*   .   C              status if true; used by scenario creator
*   .   C
```

FILE SCRCHR

```
C
C  include file SCRCHR -- screen characters
      PARAMETER sprev  = '-',   sfolow = '+',   spop   = '^',
     1          sleft  = '<',   sright = '>',   srfrsh = '&',
     2          sadd   = 'A',   sdel   = 'D',   smod   = 'M',
     3          shelp  = '?',   sinsrt = 'I',   sprint = 'P'
      PARAMETER scopy  = 'C',   sswap  = 'S',   snam   = 'N',
     5          sreloc = 'R',   squit =  'Q' ,  sendpg = 'E' ,
     6          stopmu = '/',   suse   = '{',   sstbld = '}',
     7          sjmpto = '=',   sedit  = 'T',   sclfld = 'K',
     8          srewnd = 'B',   supdat = 'U',   sverfy = 'V',
     9          smode  = 'L',   sdraw  = '*'
      PARAMETER scrchr = '-+^<>&ADM?IPCSNRQE/{}=TKBUVL*', lenscr=29
      PARAMETER nothat = '^/QE', lenoth = 4
      PARAMETER jprev  =  1 ,   jfolow =  2 ,   jpop   =  3 ,
     1          jleft  =  4 ,   jright =  5 ,   jrfrsh =  6 ,
     2          jadd   =  7 ,   jdel   =  8 ,   jmod   =  9 ,
     3          jhelp  = 10 ,   jinsrt = 11 ,   jprint = 12 ,
     4          jcopy  = 13 ,   jswap  = 14 ,   jname  = 15 ,
     5          jreloc = 16 ,   jquit =  17 ,   jendpg = 18 ,
     6          jstpmu = 19 ,   juse  =  20 ,   jstbld = 21 ,
     7          jjmpto = 22 ,   jedit =  23 ,   jclfld = 24 ,
     8          jrewnd = 25 ,   jupdat=  26 ,   jverfy = 27 ,
     9          jmode  = 28 ,   jdraw =  29
C
```

ALIAS FORTRAN INCLUDE FILES

FILE SCREEN

```
C
C  include file screen          MNUR---screen size definition
      PARAMETER WSCREN = 80
      PARAMETER LSCREN = 24
C     wscren    wide of terminal screen is characters
C     lscren    length of terminal screen in lines
C
```

FILE SENPRM

```
C  include file senprm
      parameter snmxrl=200,  snmxgp=20,  snmxgr=50
      parameter snrsun=24,  snunit=25
      parameter snread=26,   snwrit=27
C
C          scenario system parameters
C     snmxrl   maximum relation in ALIAS system
C     snmxgp   maximum number of related groups of relations
C     snmxgr   maximum number of relations in a group
C     snrsun   unit number for relsnl file
C     snunit   unit used by snok
C     snread   "
C     snwrit   "
C
```

FILE SHLIFE

```
C
C include file shlife                        FLRP/FLBG ship life data
C contains standard lifetimes for all ship classes
      PARAMETER SLIFNM='-SHLIFE.MISCJ-'
      PARAMETER SLIFFL='-SCENARIO,CLASS,LIFE,TIMUNT,DATADATE-'
      PARAMETER SLIFKY='-SCENARIO,CLASS,DATADATE:D,ENTRY_DATE:D-'
      COMMON /SHLIFE/ slcurs
      character*12 sfscen
      character*10 sfclas,sfunt*6
      integer slcurs,sflife,sfalin(17)
      integer*4 sfdatd
      equivalence (sfalin( 1),sfscen) , (sfalin( 7),sfclas)
      equivalence (sfalin(12),sflife) , (sfalin(13),sfunt)
      equivalence (sfalin(16),sfdatd)
C
      character*12 skscen
      character*10 skclas
      integer skalin(15)
      integer*4 skdatd,skdate
      equivalence (skalin( 1),skscen) , (skalin( 7),skclas)
      equivalence (skalin(12),skdatd),(skalin(14),skdate)
C
C     -- data record for SHLIFE (SHip class LIFEtime in DAYS)
C
C     slcurs     relate virtual cursor
C     s_scen     a scenrio id
C     s_clas     a ship class id
C     stlife     ship  class's standard life in years
C     s_datd     date this info was entered into relate
C
```

ALIAS FORTRAN INCLUDE FILES

FILE SNRREF

```
*   .   C  include file snrref
*   .   C     parameter lensnrref = 18/2 * snmxrl  + 12/2 * snmxrl + 1
*   .         parameter lensnrref = 3001
*   .         common /snrref/sndsid,nsets,relset(snmxgp),nsreln
*   .   C    1                 ,snrlnm(snmxrl),snrlsn(snmxrl)
*   .         integer sndsid,nsreln,nsets
*   .         character*18 snrlnm,snrlsn*12,relset*8
*   .   C
*   .   C        scenario system choice/creation data
*   .   C     nsets    number of system relation families
*   .   C     relset   id of system relation families
*   .   C     nsreln   number of relations known to scenario system
*   .   C     snrlnm   names of relations known to scenario system
*   .   C     snrlsn   scenario field key value for relation i for current
*   .   C              scenario
*   .   C
```

FILE STACK

```
C
C include file stack           stack data type storage
     PARAMETER lstack=128
     COMMON /stack/ stkflg, stkidx, stack(lstack)
     INTEGER*2 stkidx
     INTEGER*4 stack
     LOGICAL   stkflg
C       stkidx - index to top of the... stack - stack contents
C       stkflg - can be used to mean pop to top menu, when set
C
```

FILE STRNGS

```
C
C  include file strngs          DBIF string chain data storage
      PARAMETER mschn=5,mstr=3, mswstr=500,mcstr=1000
      COMMON /strngs/ strchn(mschn),intstr(mswstr,mstr)
      INTEGER   strchn,intstr
      CHARACTER str(mcstr,mstr)
      EQUIVALENCE (intstr,str)
C     strchn=string chain          str   =string pool
C     allows convenient buffering of command strings and
C     delimited text strings (DTS)
C
```

FILE TDDATE

```
C
C  Include File TDDATE
C     Defines the DATA TYPE "DDATE", meaning DatabaseDATE; it refers
C     to the integer*4 format in which RELATE returns date specs.
C
C  REPRESENTATION: INTEGER*4- Bits  2-13 (L1-12) are YEAR (ie. 1983)
C                           - Bits 17-20 (R0- 3) are MONTH (ie. 1)
C                           - Bits 21-25 (R4- 8) are DAY (ie. 30)
C  SUBTYPES:
C     Raw RELATE DDATE: a date obtained directly from RELATE tuple.
C     Clarified DDATE:  one whose unused bits are guaranteed zero.
C
C  OPERATIONS:
C     CDTODD(date_string*10 "MM/DD/CCCC") Returns(Clarified DDATE)
C     CKDATE(date_string*10,len_of_string)Returns(Boolean:T if valid
C     CKDATI(inI*2: mm,dd,yy) Returhs(Boolean: T if valid)
C     DATEP1(in/outI*2: mm,dd,yy) -- increments date by one day
C     DDATE (dummy*2) Returns(Today's date as Clarified DDATE)
C     DCLRFY(Raw DDATE) Returns(Corresponding Clarified DDATE)
C     DDTOCD(Any DDATE) Returns(date_string*10 "MM/DD/CCCC")
C     DDTOID(in: Any DDATE, outI*2: month, outI*2 day, outI*2 year)
C     IDAYS (inI*2: mm1,dd1,yy1, inI*2: mm2,dd2,yy2)Returns(I*2: 2-1)
C     JDAYS (inI*2: mm1,dd1,yy1, inI*2: mm2,dd2,yy2)Returns(I*4: 2-1)
C     IDTODD(out: Clarified DDATE, Rest are inI*2: month,day,year)
C     LMONTH(inI*2: mm,yy) Returns(I*2 number of days in that month)
C     NUMDAY(inI*2: mm,dd,yy) Returns(I*2: 1==Sunday...7==Saturday)
C     NWDATE(in: Any DDATE, inI*2: ndays) Returns(DDATE+ndays)
C     NWDATU(in: Any DDATE, inI*2: npers, inC*: per_type)Returns(DDATE)
C     NWIDAT(inI*2: mm,dd,yy, inI*2: ndays, outI*2: [mm,dd,yy]+ndays)
C  ***** D1, D2, and Dcomp below must all be clarified DDATEs *****
C  *  DCLOSR(D1,D2,Dcomp) Returns(Boolean: T if D1 is strictly     *
C  *     closer to Dcomp than D2; ie.  |D1-Dcomp| < |D2-Dcomp| )  *
C  *  DEARLR(D1,D2) Returns(Boolean: T if D1 is earlier than D2)  *
C  *  DLATER(D1,D2) Returns(Boolean: T if D1 is later than D2)    *
C  *  DEQUAL(D1,D2) Returns(Boolean: T if D1 and D2 are the same) *
C  ****************************************************************
C
C  HIDDEN OPERATIONS
C     DATEMK(inI*4: mark, outI*2: mm,dd,yy) -- distance mark to date
C     MRKDAY(inI*2: mm,dd,yy) Returns(I*4 distance mark from date)
C
C  DECLARATIONS:
      EXTERNAL  ckdate,ddtocd,cdtodd,ddate ,dclrfy,ddtoid,idtodd
      EXTERNAL  idays ,jdays ,datep1,nwdate,nwidat,nwdatu,lmonth
      CHARACTER ddtocd*10
      INTEGER*4 cdtodd,ddate ,dclrfy,jdays ,nwdate,nwdatu,mrkday
      INTEGER*4 ddzqz1,ddzqz2,ddzqzc
      INTEGER   idays, lmonth,numday
      LOGICAL   ckdate,dclosr,dearlr,dlater,dequal
C
C  STATEMENT FUNCTIONS:
      dclosr(ddzqz1,ddzqz2,ddzqzc) =
     1                 jabs(ddzqz1-ddzqzc).LT.jabs(ddzqz2-ddzqzc)
      dearlr(ddzqz1,ddzqz2)        = ddzqz1.LT.ddzqz2
      dequal(ddzqz1,ddzqz2)        = ddzqz1.EQ.ddzqz2
      dlater(ddzqz1,ddzqz2)        = ddzqz1.GT.ddzqz2
      numday(kmn,kdy,kyr) = ijint(jmod(mrkday(kmn,kdy,kyr),7J))+1
C
```

FILE TODAYC

```
C
C include file todayc           FLRP/86RP data
        INTEGER*4 TODAY, LASTDAY
        COMMON /TODAYC/ TODAY, LASTDAY
C       -- today's clarified date
C       -- maximum clarified date
C
```

FILE TRNS03

```
C
C include file trns03
C
        common /trns03/cftnam,cftdsc,cftusd,cftctr,cftctd,cftsmu
        character*8 cftnam,cftctr,cftsmu
        character*42 cftdsc
        real cftctd,cftusd
C
C          data record used to store directory info about the
C          command file currently being built for inclusion in
C          the CFLIST relation on successful build termination.
C          Twins most of rcrd03. NOT ALIGNED. DO NOT USE AS RELATE RECORD.
C
C          cftnam   command file name
C          cftdsc   command file description
C          cftusd   command file last-used date
C          cftctr   command file creator
C          cftctd   command file creation date
C          cftsmu   command file start-execution menu
```

FILE TTY

```
C
C  include file /tty/   alternative screen clear method storage
      COMMON /tty/ typtty,formfd
      CHARACTER*6 typtty
      INTEGER formfd
C
```

FILE TXTCNT

```
C
C  include file txtcnt :         MNUG---text size tracking
       INTEGER*4 NHLPLN, NDSCLN
       COMMON /TXTCNT/ NHLPLN, NDSCLN
C
```

FILE TYPEBG

```
C
C include file typebg                     FLBG---type definition data
        INTEGER NTYPE,TYPTOT(MXPERD,MXTYPE)
        CHARACTER TYPLAB*LENRLB(MXTYPE), TYPNAM*CNLEN(MXTYPE)
        COMMON /TYPEBG/NTYPE,TYPTOT,TYPLAB,TYPNAM
C       ntype   number of types defined
C       typtot  force level totals for each period, each type
C       typlab  label to be displayed under BALANCE section for type
C       typnam  name which type is cross referenced by
C
```

FILE UNTREF

```
c
c  include file untref          FILOPN/FILCLS utility storage
      common /untref/unums(99),uinuse(99)
      integer unums
      logical uinuse
c         unums  holds mpe file numbers for each logical unit
c       uinuse     true if a file has been iopeed
```

FILE UZRPRV

```
*   .   C  include file uzrprv
*   .         parameter lenuzr=61,dbalev=3,maxsec=50
*   .         common /uzrprv/uzrnam,uzrgrp,uzrlev,rdprm,wrtprm,modprm(maxsec)
*   .         common /uzrprv2/readok,writok
*   .         character*8 uzrnam,uzrgrp
*   .         integer uzrlev
*   .         logical rdprm,wrtprm,modprm,readok,writok
*   .   C     ---point alignment
*   .         integer alin07(lenuzr)
*   .         character*40   fld07(6)
*   .         equivalence (alin07,uzrnam)
*   .   C     ---module permission equivs
*   .         logical runmnu,crscen
*   .         equivalence (crscen,modprm(12))
*   .   C
*   .   C        holds user privelege info extracted from sysusr.pub
*   .   C     uzrnam   name of this user
*   .   C     uzrgrp   group user may execute from (usually home or "ANY")
*   .   C     uzrlev   user privelege summary level:1=read;2=reg;3=dba;4=super
*   .   C     rdprm    true if user may read data base
*   .   C     wrtprm   true if user may write to data base
*   .   C     modprm   (i) is true if user may execute module (i)
*   .   C     readok   true if user may read this scenario
*   .   C     writok   true if user may alter this scenario
*   .   C
```

Table 10-8. ALIAS Extra Data Segment Usage

SEGMENT	USER	PURPOSE
636	BUILDER	ID of segment which is used for communication with the default RELATE son process.
9012	DBU	Segment used for storage of cursors associated with the various RELATE son processes which the DBU starts up; i.e. DBU file management subsystem's global storage.
1	Core	Segment used by mrunp/iniprc FORTRAN routines to swap contents of Core common blocks into son process data memory. Used only as a communication segment.
101 102 103 104	DBU	Segments used for communcation [illegible]tween the DBU and the various RELATE son processes the DBU starts up using its file management subsystem.
2	SCEN	Segment used by the scenario system to store the SCENARIO field key values for each relation for the current scenario.
8001	FLRP BGRP	Temporary storage buffer used by the Force/ Battle Group Report Generators' internal page printing system.
191	ASGN	Temporary storage for partially processed tuples during the update phase of execution.
201 202	DBIF	Sequents used for RELATE son processes supervised by the DBIF.

It is crucial that modules using extra data segments, either explicitly or implicitly via the ___MEM FORTRAN utilities, not compete with one another for the same segment by specifying identical segment id numbers in the call to the getdseg intrinsic. Errors which are very difficult to trace can result from one module writing over another module's segment.

Table 10-9 presents a list of the current usage of segment id numbers.

Table 10-9. ALIAS LPRNTS Usage

LPRNT	MODULE	PURPOSE
1	LIST MEMORY	INLST and OUTLST routine diagnostics
2	HRELATE	High level RELATE utility diagnostics
3	LRELATE	Low level RELATE utility diagnostics
4	MENU SYSTEM	Input checking for MNUGEN
5	DATA ENTRY	Turns on file echo;uses DMTEST version
6	DATA ENTRY	Inhibits record adds
7	ASSIGNER	All Manual assigner system diagnostics
8	UTILITIES	filopn
10	ASSIGNER	Outbound high-level diagnostics; yard-oriented
10	ASSIGNER	routines asycls;ascmpg;asndbr;asnoui
11	ASSIGNER	Outbound job-of-interest diagnostic; asnjoi
12	ASSIGNER	Removal of hist/curr jobs from buffer before out
12	ASSIGNER	outbound processing. Routine asncln.
13	ASSIGNER	Echoes relate procedure file to screen which upd
13	ASSIGNER	updates hull numbers in ncjodat.projj. asnhul
14	ASSIGNER	High level class-oriented processing outbound.
14	ASSIGNER	Routines asgnxt ashtrb ashard astupf
15	ASSIGNER	Low level print for ashtrb; cnvt buf tc rcrd
16	ASSIGNER	Low level outbound hardwire chex; ashard
17	ASSIGNER	Low level for astupf; produces tuple images.
18	ASSIGNER	Outbound actual DB update. asndbr asodel
19	ASSIGNER	Outbound job description retrieval. aspfld asgpf
20	ASSIGNER	Outbound schedule date calculators. ascdsp ascda
22	ASSIGNER	
23	ASSIGNER	Aspred: outbound date spreading
25	SCENARIO	All scenario system debug prints
26	UTILITY	Extra data segment system (xxxMEM)

11.0 FORCE IMPACT ASSESSMENT MODULE

11.1 PURPOSE

The force impact assessment module projects future Navy force levels in terms of both raw numbers of deployable ships and deployable battle groups. It can also be used to report on past force levels if that is desired. The module is designed to permit report contents and formats to be customized to a high degree.

Sample reports of each type are shown in Figures 11-1 and 11-2.

11.2 SUMMARY OF STRUCTURE

The module is centered around two independently executable programs, each producing one of the two types of report. The programs are executed as son processes when options 2 and 3 of the Force Level Report Generator choice menu of the ALIAS Command System are chosen.

The programs are quite similar in internal structure and operation. Each reads a user-specified ASCII file, called a format control file, which specifies the contents and format of the report desired. Report contents are specified in terms of a list of ship classes for which force levels are desired (the classes may be combined into summary groups, referred to here as ship "types"). After obtaining the class list, each program searches the data base for commissionings and decommissionings of ships in the classes (as defined by construction, conversion, and reactivation jobs and decommissioning dates), for repair jobs which temporarily take ships out of the force, and for a standard service lifetime for each class. The report is then constructed and written to the device the user specifies on the User Environment Parameter menu and, optionally, to a disk file.

Figure 11-1. Sample Force Level Report Generator Output.

```
                                      POM 86 FORCE IMPACT PROJECTION
                                      BASED ON STANDARD SERVICE LIVES
                                           (ALL DATA NOTIONAL)

CALENDAR YEAR             1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999
                          ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
CV            INVENTORY      8    8    8    8    8    8    7    7    8    7    7    8    8    8
CVN           INVENTORY      5    5    5    6    6    7    7    8    8    8    8    8    8    8
CVN             PROGRAM                                                    1    1    1    1    1
                          ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
CARRIER      DEPLOYABLE     13   13   13   14   14   15   14   15   16   16   16   17   17   17

CV              IN SLEP      1    1
                          ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
CARRIER           TOTAL     14   14   13   14   14   15   14   15   16   16   16   17   17   17

BB              PROGRAM                1    1    1    1    1    1    1    1    1    1    1    1
                          ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
BB                TOTAL                1    1    1    1    1    1    1    1    1    1    1    1

CGN           INVENTORY      9    9    9    9    9    8    7    7    7    7    7    6    6    6
CG            INVENTORY     23   27   31   34   38   38   37   31   28   26   24   20   20   20
CG              PROGRAM                               2    5    9    9    9    9    9    9    9
                          ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
CRUISER           TOTAL     32   36   40   43   47   48   49   47   44   42   40   35   35   35

DDG           INVENTORY     37   37   37   38   38   38   38   38   38   38   38   38   38   38
DDG             PROGRAM                               4    8   14   19   24   25   25   25   25
                          ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
DDG               TOTAL     37   37   37   38   38   42   46   52   57   62   63   63   63   63

DD            INVENTORY     32   32   31   31   31   31   31   31   31   31   31   31   31   31
                          ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
DD                TOTAL     32   32   31   31   31   31   31   31   31   31   31   31   31   31

FFG           INVENTORY     56   56   57   57   57   57   57   57   57   57   56   52   51   51
                          ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
FFG               TOTAL     56   56   57   57   57   57   57   57   57   57   56   52   51   51

FF            INVENTORY     57   57   57   57   57   57   57   55   54   50   49   48   44   39
                          ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
FF                TOTAL     57   57   57   57   57   57   57   55   54   50   49   48   44   39

                          ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
FRIGATE           TOTAL    113  113  114  114  114  114  114  112  111  107  105  100   95   90

                          ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
GRAND             TOTAL    228  232  236  241  245  251  255  258  260  259  256  247  242  237
```

Figure 11-2. Sample Battle Group Report Generator Output.

```
                         DEPLOYABLE BATTLE GROUP PROJECTION FOR POM-86
                         BASED ON SURFACE COMBATANT REQUIREMENTS ONLY
                                     (ALL DATA NOTIONAL)

CALENDAR YEAR        1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999
                     ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----
BATTLEGROUP
-----------
   CARRIER BG           9    9    9    9    7    5    5    5    6    7    7    7    7    7
   SURFACE AG
   MARINE AF                                1    1         1                   1    1    1    1
   SUPPLY ESCORT        1    1    1    1    1    1    1    1         1
   CONVOY              10   10   10   10    8   10   10   10   10   10    9    9    9    9
BALANCE
-------
   CARRIER              4    4    4    5    7   10    9   10   10    9    9   10   10   10
   BB                             1    1    1    1    1    1    1    1    1    1    1    1
   CRUISER              7    7    7    7    7    9    9    7    8    7    5    3    3    3
   DDG
   DD                   2    2    2    2         2   10    2    9    6
   FFG
   FF                  33   33   34   34   40   40   50   40   47   37   31   26   21   16
```

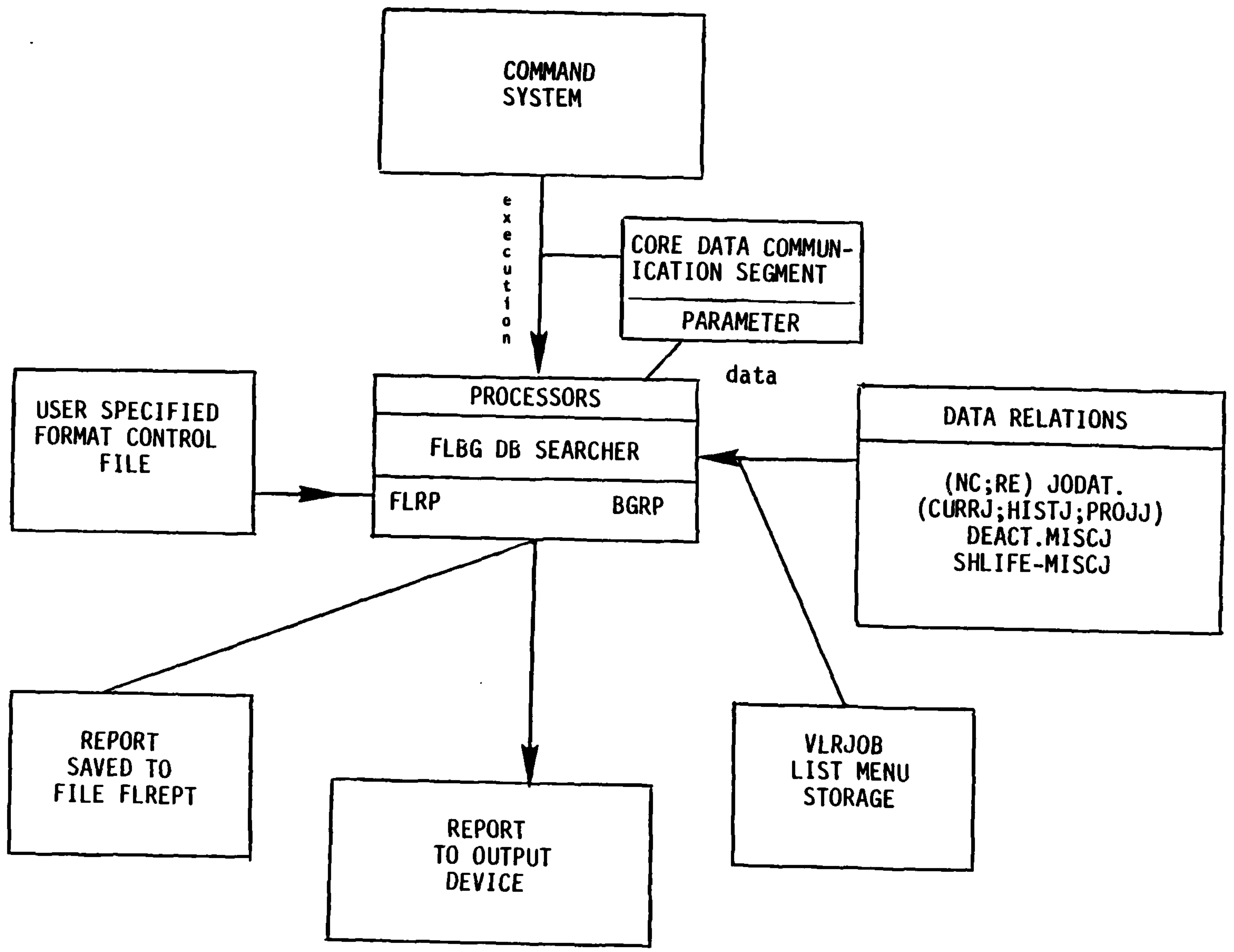


Figure 11-3. Structure of Force Analysis Modules

Since the actual numbers on the reports depend only on the contents of the data base, and particularly on schedules, force impact reports will always reflect the latest data available for a scenario.

The programs' only direct interaction with the user is the prompt for the name of the input format control file; all other control values, such as the dates specifying the period for which the report is desired, are taken from the settings of the Command System's Force Level Parameters Menu.

Although FLRP and BGRP are completely separate in terms of executability, they do share a good deal of source code, particularly the data base search logic.

This structure is summarized in Figure 11-3.

11.3 INPUTS AND OUTPUTS

This section will describe FLRP and BGRP inputs and outputs in more detail. They will be discussed here as though they were a single program, since the structure of their inputs and outputs is identical.

The sole outputs of the module (with the exception of prompts and error messages) are the reports themselves, in the general form shown in Figures 11-1 and 11-2. These reports are always written to the user's default hard copy output device (the one specified on the User Environment Parameters menu), and may also be saved in a disk file for editing by the user and subsequent re-printing.

Six types of input are required. In order of first use by the module, these are the System Core process data swap with its parameter menu contents, the name of the format control file from the user, the contents of the Out of Force Repair Jobs List Menu associated with the Force Level Parameters Menu in the Command

System, the contents of the format control file, scenario key field values for various relations as supplied by the scenario system, and the contents of various data base relations for the user's current scenario.

11.3.1 Core Swap Data Used

As described in Section 8, the System Core will swap out the contents of many of its key global arrays into an extra data segment just before activating a son-process module, if the developer desires. The data may then be read into identical global arrays in the son by a call to the iniprc utility.

FLRP and BGRP make use of this facility to obtain the current values of the variables appearing on the Force Level Parameter Menu in the Command System. These variables hold many of the control values for FLRP/BGRP program execution. In order to minimize the number of routines in which the /pvalue/ array must be included, the values are moved by the flinit routine into a common block called /flvalu/ which is used only by FLBG/FLRP.

A sample of the parameter menu is shown in Figure 11-4. In the order in which they appear, the use of the parameters is:

1) KEEP REPORT: If set to YES, the output of FLRP/BGRP will be saved in a file in the log-on group called flrept. If flrept already exists at module run time, the user is prompted for an alternative name.

2) REPORT START: The first day of the first period the user is interested in. Ships reaching final retirement before this date will never appear on a report regardless of the contents of the format control file. If the user specifies a date that is not the first day of its period, the date is moved back to the first day.

3) REPORT END: The last day of the last period the user is interested in. Determines the number of periods in conjunction with REPORT START.

4) RETIRE SHIPS BY: If DATE, the data base search logic will look for specific retirement dates for each ship in the deact.miscj relation for those ships not already retired as of the day the report is being run. If it

Figure 11-4. Command System Parameter and List Menus Serving the Force Module

```
Menu is FLREPT              * ALIAS COMMAND SYSTEM *   Scenario is DEMO
---------------------------------------------------------------------------
               FORCE LEVEL AND BATTLEGROUP REPORT GENERATOR PARAMETERS
 1.   KEEP REPORT ON-LINE        = YES          (YES,NO)
 2.   REPORT START DATE          =  1/ 1/1986   (MM/DD/YYYY)
 3.   REPORT END DATE            = 12/31/2005   (MM/DD/YYYY)
 4.   RETIRE SHIPS BY            = LIFE         (LIFE,DATE)
 5.   TIME PERIOD LENGTH         = CALYR        (DAY,WEEK,MONTH,QTR,CALYR)
 6.   IN FORCE DAY               = END          (BEGIN,END)
 7.   PROGRAM MILESTONE          = APPROP       (APPROP,AWD,DELIV)
 8.   OUT OF FORCE REPAIR JOBS   = LIST         (ALL/LIST)

  COMMAND:
```

```
Menu is FLREPT              * ALIAS COMMAND SYSTEM *   Scenario is DEMO
---------------------------------------------------------------------------
             REPAIR JOBS THAT REMOVE A SHIP FROM FORCE DURING EXECUTION
---------------------------------------------------------------------------
    1.   REFUEL                            3. * SLEP
    2.   REPAIR                            4.   TESTRE
---------------------------------------------------------------------------

  COMMAND:
```

cannot find a date specification there, the standard service life for the class, in combination with the amount of service the ship has seen (not including periods of deactivation), will be used to determine the final retirement date. If LIFE, then the standard service life will be used for all ships not retired by the day the report is run.

5) TIME PERIOD LENGTH: The time units the period of interest should be measured in. A variety of choices is available, but no report may span more than 20 periods.

6) IN FORCE DAY: There must be a rule to determine whether ships retiring in the middle of a period are in the force for that period or not. If this parameter is END then they are not; if BEGIN then they are.

7) PROGRAM MILESTONE: In the format control file the user may specify that the force availability for ships of each type be separated into multiple lines, or "programs", based on a milestone in each ship's construction process. The purpose of this is to let the user see, for example, the relative impact of ships already built compared to ships in the POM compared to those in the EPA. The user will specify start dates for each era or program in the control file; those dates will be compared to the construction/conversion/-reactivation milestone date for each ship specified here. Thus if APPROP is chosen, ships appropriated after the first day of the POM era will be placed in POM lines on the report.

8) OUT OF FORCE REPAIR JOBS: This is a gate to a list menu whose role is discussed in Section 11.3.3.

11.3.2 Format Control File Name

Since format control files are just standard editor files containing a particular syntax, a large number and variety of them can exist. For this reason the user is prompted for the name of the file he wishes to use, rather than limiting the choice to a small selection of values on the parameter menu. Most public format control files are maintained in the .fmtfil group.

11.3.3 Out of Force Repair Jobs List Menu

A sample of this list menu is shown with the parameter menu in Figure 11-4. In this menu the user specifies which kinds of repair job will cause a ship to be temporarily out of the force level for purposes of a force impact study. The contents of the candidates list on this menu is managed by the RE_JOB_TYPES screen of the DBU, so the user can cause any or all of the legal repair job type code names to appear on the list. He can turn any, all, or none of them on (only SLEP is on in the sample). If none are on then no temporary removals will be effected.

The out-of-force-repair-job concept was implemented to deal with SLEP jobs in particular; it has been common to consider a carrier in SLEP as not in the force level in force impact studies in the past.

11.3.4 The Format Control File

Sample Force Level and Battle Group Report format control files are shown in Figures 11-5 and 11-6. Syntax rules for the files are discussed in the ALIAS User's Guide, and will be discussed here only as appropriate.

Both files are keyword-oriented; that is, the logic which reads them identifies the type of data appearing on a given line, and the actions to take on that data, according to keyword commands which appear as the first word on the line. Where a line must be longer than 72 characters (the maximum width allowed) then continuation lines beginning with a "+" may follow it.

In both files the TITLE keyword must appear before other keywords; the same applies to PRGLB in the Force Level file.

Note that although the Force Level format control file is larger, the Battle Group file has a larger variety of keywords and is more complex. Here key word lines must appear in groups and strictly in the order TYPE, FUNCTION, BGROUP, MAKEUP. The user specifies the TYPEs of ships available in terms of classes; then the FUNCTIONs each type can perform; then the battle groups desired and the target number and priority for each (note that a

Figure 11-5. Sample Force Level Report Format Control File

```
% THIS IS A FORCE LEVEL REPORT FORMAT CONTROL FILE
% ANY LINE BEGINNING WITH % IS CONSIDERED A NOTE AND IGNORED
%
%    The next two lines tell FLRP to split the force level
%    into two lines for each class, based on ship approp date
PRGLB   Inventory,1/1/1900
PRGLB   Program  ,10/1/1986
%
%    The TITLE lines give the title that will be printed
%    (centered) on the top of each report page
TITLE POM 86 Force Impact Projection
TITLE Based on Standard Service Lives
TITLE (All Data Notional)
%
%    Start the report specification.  BTOT lines tell FLRP to
%    start keeping a running total, ETOT where in the body
%    to print the total; last two words on ETOT lines are
%    the left and right labels actually printed.  Label
%    on BTOT line and first one on ETOT for internal FLRP use.
%    EITOT is like ETOT except specifically designed for
%    subtotals; it ensures no page feed happens in the
%    middle of a type being printed.
%    TYPE lines specify ship types by
%    giving the names of all the classes in the type.

START
BTOT grand
BTOT subn
TYPE SSBN,      SSBN-726,SSBN-640,SSBN-627,SSBN-616,
+               SSBN-611,SSBN-610,SSBN-609,SSBN-601,SSBN-599,
+               SSBN-598
ETOT subn,SSBN,TOTAL
BTOT sub
TYPE SSN,       SSN-688,SSNX,SSN-21,SSN-575,SSN-578,SSN-585,SSN-594,
+               SSN-597,SSN-637,SSN-671,SSN-685
ETOT sub ,SSN,TOTAL
%
%    note job line causes carriers in SLEP to be printed
%    they do not appear in the deployable total
BTOT carrier
BTOT dcarrier
TYPE CV,        CV-41,CV-59,CV-63,CV-67
TYPE CVN,       CVN-65,CVN-68
EITOT dcarrier,CARRIER,DEPLOYABLE
JOB CV,         IN SLEP,SLEP,       CV-41,CV-59,CV-63,CV-67
ETOT carrier,   CARRIER,TOTAL
%
BTOT bb
TYPE BB,        BB-61
ETOT bb,        BB,TOTAL
```

Figure 11-5. Sample Force Level Report Format Control File

```
%
BTOT cruisers
TYPE CGN,       CGN-25,CGN-36,CGN-38,CGN-35,CGN-9
TYPE CG,        CG-16,CG-26,CG-47
ETOT cruisers,  CRUISER,TOTAL
%
BTOT ddg
TYPE DDG,       DDG-2,DDG-37,DDG-51,DDG-993
ETOT ddg,       DDG,TOTAL
%
BTOT dd
TYPE DD,        DD-945,DD-963
ETOT dd,        DD,TOTAL
%
BTOT frigate
BTOT ffg
TYPE FFG,       FFG-1,FFG-7
ETOT ffg,       FFG,TOTAL
BTOT ff
TYPE FF,        FF-1037,FF-1040,FF-1052
ETOT ff,        FF,TOTAL
ETOT frigate,   FRIGATE,TOTAL
%
BTOT amphibs
TYPE AMPHIBS,   LHD-X,LSD-41,LSD-49,LCC-19,LHA-1,LHD,LHD-1,
+               LKA-113,LPD-1,LPD-4,LPH-2,LSD-28,LSD-36,LSD-41,
+               LSD-49,LST-1179
ETOT amphibs,   AMPHIBS,TOTAL
%
BTOT mine
TYPE MINE CM,   MCM-1,MSH-1,MSO-422,MTS
ETOT mine,      MINE SHIPS,TOTAL
%
BTOT aux
TYPE AUXILIARY,AD-37,AD-41,AE,AE-21,AE-23,AE-26,AF-58,AFDM,
+               AFS-1,AG,AK-286,AO-143,AO-177,AO-187,AO-51,
+               AOE,AOE-1,AOR-1,AR,ARDM-4,ARS-50,AS-19,AS-31,AS-33,
+               AS-36,AS-39,ASR-21,ATF-166,ATS-1
ETOT aux,       AUXILARY,TOTAL
%
BTOT T-SHIPS
TYPE T-SHIPS,   T-ACS,T-AG,T-AGOS-1,T-AO-187,T-ARC-7,T-AVB,TAGOS-1,
+               TAH-X,TAO-187
ETOT T-SHIPS,   T-SHIPS,TOTAL
%
ETOT grand,     GRAND,TOTAL
STOP
```

Figure 11-6. Sample Battle Group Report Format Control File

```
% ALIAS BATTLE GROUP REPORT FORMAT/CONTENTS DEFINITION FILE
% format is: title; start; type; function; bgroup; makeup; end
% title line has titles for report
% start line indicates start of processing
% type line indicates ship classes making up a type
% function line lists types which can perform a function, in
%  order of preference
% bgroup  describes battle groups to be made up
% makeup  describes which functions each battle group requires
%
TITLE Deployable Battle Group Projection For on POM-86
TITLE Based on Surface Combatant Requirements Only
TITLE (All Data Notional)
%
START
%
% type format similar to force level report: name,label,class list
%
TYPE CARRIER,   CARRIER,    CV-41,CV-59,CV-63,CV-67,CVN-65,CVN-68
TYPE BB,        BATTLESHIP,BB-61
TYPE CRUISER,   CRUISER,    CGN-25,CGN-36,CGN-38,CGN-35,CGN-9,CG-16,CG-26,CG-47
TYPE DDG,       DDG,        DDG-2,DDG-37,DDG-51,DDG-993
TYPE DD,        DD,         DD-945,DD-963
TYPE FFG,       FFG,        FFG-1,FFG-7
TYPE FF,        FF,         FF-1037,FF-1040,FF-1052
%
% function format is name,list of types which can perform it
%  in order of preference
FUNCTION CRUISER, CRUISER,BB
FUNCTION CARRIER, CARRIER
FUNCTION DDG,     DDG
FUNCTION DD,      DD
FUNCTION FRIGATE, FFG,FF
%
% bgroup format is name,output label,priority,target level,
%  begin date this defn takes effect, end date this defn effective
BGROUP CVBG,CARRIER BG,1,15,1/1/1900,1/1/2111
BGROUP SAG,SURFACE AG, 3, 4,1/1/1900,1/1/2111
BGROUP MAF,MARINE AF,  2, 2,1/1/1900,1/1/2111
BGROUP ESC,SUPPLY ESCORT,4,10,1/1/1900,1/1/2111
BGROUP CON,CONVOY,     5,10,1/1/1900,1/1/2111
%
% makeup format is battle group name, function, # reqd, func, #reqd
MAKEUP CVBG,    CARRIER,1,CRUISER,1,DDG,4,DD,2,FRIGATE,4
MAEKUP SAG,     CRUISER,2,DDG,2,FRIGATE,2
MAKEUP MAF,     CRUISER,2,DDG,2,DD,8,FRIGATE,10
MAKEUP ESC,     DDG,1,DD,1,FRIGATE,2
MAKEUP CON,     DD,1,FRIGATE,4

%
STOP
```

given target applies through the date given on the BGROUP line); then the functions which are required to MAKEUP each group.

The Force Level format file specifies only TYPEs, since the force report is raw numbers available by type, but also has the capabilitiy to total and subtotal types. A stack-like logic is used in which the user "pushes" another total onto the list of those FLRP is making up with a BTOT line, and "pops" it off (causing it to be printed) with an ETOT or EITOT line. Types and totals will appear on the output in the order in which they appear in the file.

Likewise, battle groups and type balances appear on the Battle Group report in the order in which they are named in the Battle Group report control file.

The necessity of letting the user specify both the contents and order-of-output of both types of report was what prompted the use of format control files. These are non-standard ALIAS constructs because they require the user to know some syntax, use the editor, and operate in a fairly unsupervised and unaided fashion. However, a method of report specification relying only on standard facilities such as list menus would have been very clumsy and limiting.

11.3.5 Scenario Key Field Values

FLRP and BGRP make use of scenario system services via the DBIF and the contents of the cursen array in the /scenar/ common block (i.e. in the usual fashion) when constructing search keys for retrievals from the relations.

11.3.6 Relations

FLRP and BGRP read the contents of ten relations.

Vlrjob.mnurel holds the contents of the Out-Of-Force-Repair-Jobs list, and is searched by the programs via a call to the liston utility routine in order to recover the names of any job types which are "on". Liston also reads the lccref.mnurel cross referencing relation, which is opened by iniprc. These relations are managed by the Command System and need be of no great concern.

The ncjodat.histj, ncjodat.currj, and ncjodat.projj relations hold schedules for historical, current, and projected new construction, conversion, and reactivation jobs. These schedule records are vital data for this module, since they indicate the number of ships that enter the force over time and the timing of each entry.

The .histj and .currj versions of the relations can contain both actual and projected schedule records for a given ship, and for multiple data dates. The rule used by this module for selecting which single record to use for a given ship is based solely on datadate: the record with the latest datadate is used. The intuition behind this is that regardless of whether the DATETYPE field indicates the data is actual or projected, the record with the latest datadate is most likely to contain the Navy's best guess as to the commissioning of a given ship. Note that if no commissioning date is given, the delivery date is used instead, and that if no delivery date is given, the ship is simply ignored.

The rejodat.histj, rejodat.currj, and rejodat.projj relations form a similar structure containing repair job schedule records. They are searched with similar rules, but their data is less central to force impact studies since they are consulted only for job types which are "on" in the Out-Of-Force-Repair-Jobs list menu.

The deact.miscj relation holds actual and projected ship deactivation dates (note that a known deactivation date will appear here regardless of which new construction schedule relation the given ship's activating job record appears in). This relation is searched for every activation found in the ncjodat relations. This date is used if it is earlier than the date the report is being run on (i.e., the ship has already retired), or if the user has chosen DATE for the RETIRE SHIPS BY parameter.

The shlife.miscj relation must have one record for each ship class which appears in a report (this condition is met automatically as long as the DBU is used for data base maintenance). The record gives the standard service life for ships of that class, which is used to estimate a given ship of that class's retirement date if no deactivation date projection can be found for the ship in deact.miscj (or if the LIFE option is chosen on the parameter menu).

The relations are accessed through standard DBIF calls. FLRP and BGRP do require that a number of special indexes exist for the relations to support their POINT-oriented search logic.

11.4 DATA STRUCTURES

11.4.1 Data Structures Used by Both Programs

Both FLRP and BGRP use the data relations discussed in the preceding section, the relevant format control file, an extra data segment, a direct-access ASCII file, and a number of common blocks.

The only additional thing to be noted about the relations is that both the repair and the new construction schedule relations are each opened twice. The construction relations are opened twice on the same index (and naturally on different cursors/partitions since the DBIF is used) because after a construction/conversion job is found for a given ship, a search must be conducted for possible reactivation jobs. The two searches would interfere with one another if conducted through the same cursor. The repair relations are opened on different indexes to support two different kinds of searches.

One of the problems which had to be solved for FLRP in particular was the matter of appropriate location of page feeds as the report file is printed. It is desirable to have all the lines in a particular subtotal group appear on the same page (the assumption here is that in reports with multiple lines for a TYPE totaling will be specified over the lines, as is done in the sample in Figure 11-5). To ensure this, output lines are sent to a holding buffer rather than directly to the output file. The contents of the buffer are flushed to output only when an ETOT line is encountered. This buffer is actually an extra data segment (it was originally a common block, but memory limits required use of the segment). Transfer of lines to and from the segment is managed by calls to the ___mem utilities (e.g. getmem).

Rather than being sent directly to the unit which is to produce the hard copy output (a problem if the output device is not spooled, since exclusive access will be required, thus tying the device up), report lines are instead sent (from the extra data segment buffer) to a sequential-access ASCII file. When the report is complete the contents of the file are read and sent to the output device in a tight loop. If the user has specified that the report be kept in a disk file, then the given file is just saved rather than being deleted.

Table 11-1 presents an annotated listing of the common blocks used by the Force Impact module programs. These common blocks form the principal working data structure for computational purposes.

The most important block used by both programs is contained in the fltabls.incl include file. A large array, indexed by ship classes, period, and programs (FLRP format control PRGLB keyword)

TABLE 11-1. Include Files Used By the Force Report Generators.

FILENAME	PURPOSE
BGPMTR	FORTRAN Parameters defining battle group report generator capacity limits, e.g. maximum number of functions definable.
BGTITL	Title of the battle group report section now being printed (either BATTLEGROUP or BALANCE).
FLCLASS	List of ship classes named by the user on TYPE lines in the format control file, i.e. list of classes whose ships are to be retrieved from the data base and their force increments computed.
FLCONCH	FORTRAN parameters defining key words acceptable in the force level report format control file.
FLCONS	Relation names, index lists, field lists, and record buffers for opening and retrieving data from all three of the ncjodat.(projj currj histj) new construction schedule relations.
FLDECM	Same as FLCONS but for the deactivations relation deact.miscj.
FLHEAD	User-specified titles to appear on the report and the period-labeling portion of the page header.
FLIOC	FORTRAN io unit assignments for flrp and bgrp. Included are the format control file unit number, the hard-copy output unit, and the save-to-file unit.
FLJLST	List of force-affecting repair job types, i.e. those repair jobs which cause a ship to be temporarily removed from the force level while undergoing the jobs.
FLPAGE	Line numbers and record used to manage/communicate with the extra data segment in which output is temporarily stored by flbg's full-page printing subsystem.
FLPERD	Number of periods being considered this run and an array holding the date of the first day of each period.
FLPMTR	FORTRAN parameters defining flbg capacity limits, e.g. maximum number of classes specifiable on TYPE

TABLE 11-1. Include Files Used By the Force Report Generators.

FILENAME	PURPOSE
	lines.
FLRJOB	Like FLCONS but for the repair job schedule relations rejodat.(projj currj histj).
FLTABLS	The array constructed by the data base search algorithm, containing the number of deployable ships in each class by program type in each period.
FLTOTL	The arrays holding the running totals which are output on force level reports when an ETOT or EITOT keyword is found in the control file.
FLVALU	The values of the variables shown as parameters on flbg's command system parameter menu, as extracted from the /pvalue/ array by the flinit routine.
FUNCBG	Names of battle group functions, as defined on FUNCTION lines, and array locations of the TYPEs which can perform those functions.
GROUPBG	Information describing the battle groups defined as desired on the BGROUP lines of battle group format control files. See also MKUPBG.
INCPAR	System Core (command system) capacity defining FORTRAN parameters, mainly used here to specify terminal input line length.
IOC	Standard ALIAS FORTRAN io unit numbers; mainly in and iout used here.
LPRNTS	Array of diagnostic print switches.
MKUPBG	Linked list of functions (and amount of each) required to make up each battle group.
READC	Line number of last line read from input file.
SCENAR	Scenario system information; current scenario name and scenario field key value for queries on each relation opened using the DBIF.
SHLIFE	Like FLCONS but for the ship class standard lifetime specification relation shlife.miscj.

TABLE 11-1. Include Files Used By the Force Report Generators.

FILENAME	PURPOSE
TDDATE	The ALIAS date data type/date utility system declarations file.
TODAYC	Today's date in ddate form and the maximum possible ddate.
TYPEBG	Storage for availability figures by period for each TYPE defined on bgrp format control file TYPE lines; also labels and type names.

is declared here. The raw results of the data base search (number of ships of each class available in each period, by era of construction) are placed here for refinement into the final report format.

11.4.2 Key BGRP Common Blocks

In addition to these data structures, program BGRP makes use of several additional important common blocks. The /groupbg/ block contains the name, priority, etc. for each requested battle group as well as the actual numbers which are computed to be achievable. The /mkupbg/ block holds an array, managed as a linked list, which lists the "functions" which make up each battle group and their numerical requirements. /funcbg/ contains information which supports cross-referencing between function names and lists of ship "types" which can perform the functions, and /typebg/ contains the number of each type available in each period (summarized from the contents of /fltabls/).

11.5 PROCESSING LOGIC

11.5.1 FLRP

Table 11-2 lists the routines which comprise the FLRP program (not including general-purpose ALIAS FORTRAN utilities or the routines in the DBIF) and indicates which source file family they reside in. Table 11-3 provides a complete annotated listing of all the routines in FLRP and BGRP. See Section 11.8 for complete abstracts of routines. A calling tree diagram for FLRP appears in Figure 11-7.

This section will summarize the logic of the program.

FLRP initialization includes retrieval of the data placed in the swap segment by the Core, retrieval of the list of Out-Of-Force-Repair-Jobs, prompting for the name of the format control file, and opening of all the relations required. This activity is supervised by the flinit routine.

TABLE 11-2. Alphabetical Listing of Routines in FLRP A Program

ROUTINE	HOST FILE
FFLTBL	FLBGxxx
FLADPG	FLRPA
FLBRPT	FLRPA
FLBUGI	FLBGxxx
FLCHK1	FLRPA
FLCHK2	FLBGxxx
FLCHK3	FLBGxxx
FLCLAS	FLBGxxx
FLCLOS	FLBGxxx
FLDECR	FLRPA
FLGLIF	FLBGxxx
FLINCL	FLBGxxx
FLINCR	FLBGxxx
FLINIT	FLRPA
FLJOB	FLBGxxx
FLNPER	FLBGxxx
FLNXTP	FLBGxxx
FLPARS	FLBGxxx
FLPDAY	FLBGxxx
FLPMTH	FLBGxxx
FLPQTR	FLBGxxx
FLPRGN	FLBGxxx
FLPRNT	FLBGxxx
FLPROC	FLRPA
FLPWEK	FLBGxxx
FLPYER	FLBGxxx
FLRDCN	FLBGxxx
FLRDLN	FLBGxxx
FLREPT	FLRPA
FLRPGN	FLBGxxx
FLTYPE	FLRPA
FLWRIT	FLBGxxx
FLWTOP	FLRPA
FNDPRD	FLBGxxx
GETJOB	FLBGxxx
GETLIF	FLBGxxx
PAR2LN	FLBGxxx
PAR3LN	FLBGxxx
READFL	FLRPA
SKIPFL	FLBGxxx

Table 11-3. Annotated List of Force Level and Battle Group Report Generator Routines

ROUTINE	PURPOSE
BGADPG	Adds an availability-row line (i.e. a set of numbers by periods) to the line buffer (extra data segment).
BGBRPT	Like FLBRPT but more complex, this routine is the executive for actual production of the battle group report text. It processes the format control file and prepares the data structure for report production. Then calls BGGET and BGMRPT for output construction.
BGFUNC	Processes a battle group format control file function line. Principal output is the fnmlst and fdefine arrays, a list of defined function names and an array which allows cross referencing from the name to TYPE storage array elements.
BGGET	Computes battle group availability when /fltabls/ and format control file processing is complete. Outer loop is over periods, inner over battle groups in order of priority. Groups are made up with provisional decrementing of the available TYPE pools, which is committed when a group is fully constructed.
BGINIT	Zeros relevant arrays.
BGMKUP	Processes a MAKEUP line from the battle group format control file. Output is a linked list in the BGMKUP array which specifies which functions, and how many of each, are required to makeup the given battle group.
BGMRPT	Actually writes lines of the computed report to the line buffer (extra data segment).
BGPROC	Similar to FLPROC, makes the first pass through the battle group format control file in order to construct the list of ship classes of interest which ffltbl requires. Also reads and stores TITLE lines.
*BGREPT	Main program unit and chief executive for the BGRP battle group report generation program. Calls other high-level routines to do the actual work.
BGSETV	Increments a row array with a given value for the period between two given dates. Used to set the target number of each battle group; this target can

Table 11-3. Annotated List of Force Level and Battle Group Report Generator Routines

ROUTINE	PURPOSE
	vary over time.
BGTYPE	Processes TYPE lines from the battle group report format control file. Output is the TYPTOT array, which is the number of ships of all the classes named on the type line which are available in each period.
BGWRIT	Similar to FLWRIT, now unused.
BGWTOP	Writes titles and section header lines to the sequential storage file when a new page is called for.
BTLGRP	Processes BGROUP lines from the format control file. Output is the /groupbg/ common block describing the battle groups and their target and achieved amounts.
FFLTBL	Executive for the search of the data base for data on commissionings and decommissionings of classes of interest. The bulk of the logic is acutally in this routine. It searches every construction and repair relation for jobs done on ships in the classes specified in the format control file, and appropriates increments the /fltabls/ data structure. See the text for more information on the algorithm.
FLADPG	Writes an array line to the text buffer (extra data segment).
FLBRPT	Executive for actual creation of the Force Level report output. Re-reads the format control file and constructs output lines based on its directives and using the data created by FFLTBL.
FLBUGI	A service routine called to read a file line; written in response to a compiler bug which caused legal code using the usual utilities to be uncompilable.
FLCHK1 FLCHK2 FLCHK3	These three logical function utilities take one, two, and three pairs of string arguments, respectively and return whether or not they are equal. They are used mostly in checking tuples retrieved from relations to see if they have the proper key values.
FLCLAS	Subsidiary of FLPROC which supervises construction of

Table 11-3. Annotated List of Force Level and Battle Group Report Generator Routines

ROUTINE	PURPOSE
	the list of classes of interest. Takes a TYPE line, makes additions to /flclass/.
FLCLOS	Closes all files and relations. The act of closing the report output file starts printing if the "file" is in fact a spooled device.
FLDECR	Like FLDECR except subtracts one from the elements of the array. Used to remove a ship from the reported force for the periods it is out for repairs, if any.
FLGLIF	Given a construction/deactivation/reactivation history for an individual ship and its standard lifetime, calculates the ship's projected final retirement date.
FLINCL	Takes a character variable and a list in the form of an array and adds the variabl's contents to the list if it is not already on the list. Used to, e.g., manage additions to the list of classes of interest.
FLINCR	Increments elements of a row of the main /fltabls/ array (correponds roughly to a row on the report output) by one for those elements representing the period between two given dates. Essentially, adds a ship to the reported force for its lifetime.
FLINIT	Main initialization routine for both FLRP and BGRP, with source code in recomp.src (since it must read the /pvalue/ data structure. Transfer parameter values from /pvalue/ into /flvalu/, prompts for and opens the format control file, opens the output file and/or device, and opens all the relations which will be involved in the data base search.
FLJOB	Processes a JOB line from a force level report output control file, producing a line for the output report. Constructs a list of classes, taken from the input line, and calls GETJOB to find out how many ships of each class were out for each job in each period. Then formats and sends the output.
FLNPER	Using the start and end dates of interest and the time units specification from the parameter menu, figures out how many periods there are in the exercise.

Table 11-3. Annotated List of Force Level and Battle Group Report Generator Routines

ROUTINE	PURPOSE
FLNXTP	Figures out what report period (output column) a date falls in by comparing it with the array of period-start dates set up during initialization.
FLPARS	Parses an input string consisting of a list delimited by commas into individual elements, placing the elements in an array.
FLPDAY FLPMTH FLPQTR	A series of date utilities, written before the gdatep /gpern/fddate general purpose period utilities, which figure out how many periods of the given type there are between two given dates. Also, fills in the array of start dates of each period.
FLPRGN	Integer function used to find out which program line a given job will fall on based on its basis date. I.e. will a job be, e.g., inventory (1) or program (2).
FLPRNT	During report construction, output lines are first stored in an extra data segment until a complete set of lines (i.e. including all associated totals) can be sent; the set of lines are then printed to a sequential holding file (the one the report will be saved in if the user has requested a save on disk). This routine rewinds the holding file and writes the report to the output device when report construction is completed.
FLPROC	Conducts the first read of the format control file, the object of which is construction of a list of the ship classes the user is interested in (/flclass/). This list is required by ffltbl. Also looks for the PROGRAM keyword lines to find out how many lines to split each classes members into, and stores the user-specified TITLE lines.
FLPWEK FLPYER	See FLPDAY above.
FLRDCN	Utility for reads of the format control file. Calls FLRDLN and counts the number of lines returned.
FLRDLN	Reads a line from the file open on a given FORTRAN unit number.

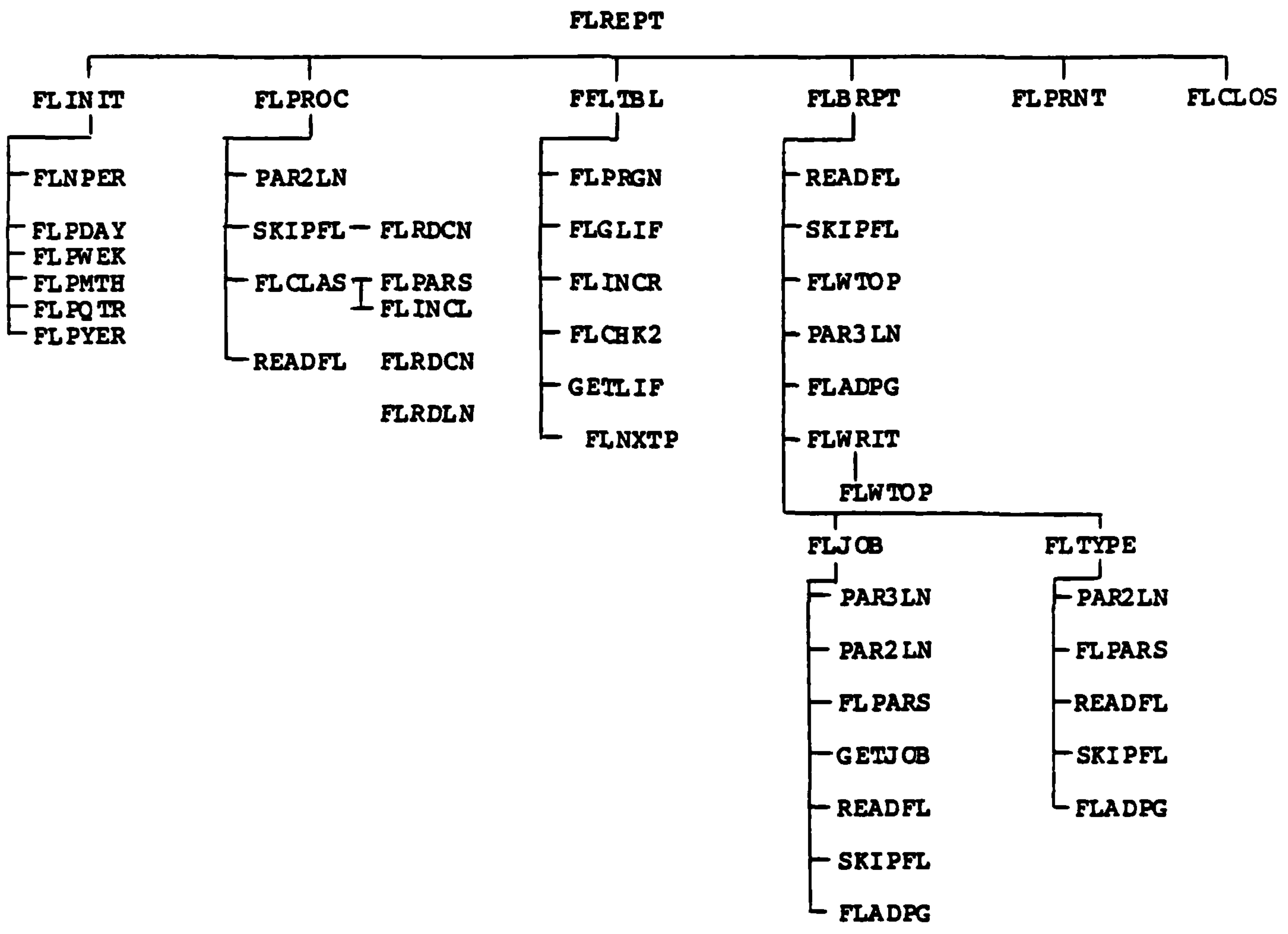
Table 11-3. Annotated List of Force Level and Battle Group Report Generator Routines

ROUTINE	PURPOSE
*FLREPT	Main program unit for FLRP, the force level report generator. An executive which calls six high-level routines to do the work of producing the report.
FLRPGN	Obsolete version of FLPRGN.
FLTYPE	Processes a format control file TYPE line during actual report constuction, converting it into output lines. Uses the list of classes to search /fltabls/ for rows of deployable ships in each period, summing these rows, and writing out the result when finished.
FLWRIT	Writes the contents of the line buffer to the sequential-access output file.
FLWTOP	Writes the title and period header lines to the line buffer/extra data segment, i.e. starts a new page.
FNDPRD	Given a date, finds which period it belongs in. Obsolete.
GETJOB	Given a class name and job type code, searches the repair schedule relations for instances of that job on that class. Increments a row-array for each one found for the period the given ship is undeployable.
GETLIF	Retrieve (from shlife.miscj) and convert to days the standard lifetime of the ships in a given class.
GMAKUP	Returns a record from the linked list of functions comprising a given battle group in the /mkupbg/ common block.
PAR2LN	FBLG utility which takes a line of elements separated by commas, separates off the first element, and returns that element and the remainder of the line. Used to, e.g., extract labels from format control input lines.
PAR3LN	Like PAR2LN except returns the first two elements and the remainder.
READBG	Reads a line from a battle group format control file and decodes its keyword. Strips off the keyword and returns the rest of the line.

Table 11-3. Annotated List of Force Level and Battle Group Report Generator Routines

ROUTINE	PURPOSE
READFL	Reads a line from the FLRP format control file and decodes its keyword. Strips the keyword from the line and returns the remainder to the caller.
SKIPFL	Reads and discards lines from the format control file until a line with a START keyword is found.

Figure 11-7. FLRP Calling Tree Diagram



Flproc is then called to read through the format control file in order to construct a list of the classes of interest for the report, as specified on TYPE lines. Flproc also reads and stores away the user-specified report title and the program-era specifications given on any TITLE and PRGLB lines.

With this information in hand the search of the data base can be conducted. This search is managed by the ffltbl routine. The logic of the routine is built around (or inside) a loop over the list of classes of interest, as shown in Figure 11-8.

For each class, its standard service life is first retrieved. Then all jobs on ships of that class are retrieved from the ncjodat.(histj currj projj) relations, one relation at a time, from historical to projected. For each job found, an additional search of the ncjodat relations is made to see if there are any subsequent reactivations. Also, a search is made for a specific retirement date for the given ship. The proper program line(s) of a holding buffer are incremented for each period when the given ship was active. When all the activating jobs have been processed, a loop over the repair job relations retrieves all repair jobs of interest for the given class, and the holding buffer is decremented in the appropriate periods. When all processing is complete for the given class, the holding buffer is moved into the /fltabls/ storage array.

This step consumes almost all of the large amount of execution time required by FLRP. The low apparent rate of progress is caused by the large number of data base queries which are required; these queries are each relatively time consuming because of RELATE response time limitations.

Once the raw per-period force availabilities are computed by class, the actual output report can be constructed. This process is supervised by the flbrpt routine, which rewinds and

Figure 11-8. FFLTBL FLOW OF EXECUTION SUMMARY

```
FOR EVERY CLASS OF INTEREST
     Get standard life of ships in class

     FOR EVERY NC RELATION (histj to projj)
          Find next job of interest in class
          Find associated decommissioning date
          Compute force level increment
          Look for reactivations
               When found, note life added and
               look for later decommissionings
     NEXT NC RELATION

     FOR EVERY RE RELATION (histj to projj)
          Look for jobs in this class that are
          job types turned "on" on list menu
               When found, decrement force level

     NEXT RE RELATION

     SAVE INFO FOR THIS CLASS INTO /FLTABLS/

NEXT CLASS
```

re-reads the format control file, now processing every keyword line (except TITLE and PRGLB). In particular, for each TYPE line encountered, a list of the classes in the type is constructed and the per-period availability of each class is extracted from /fltabls/ and summed into a holding array. This array can be thought of as being composed of the rows which appear on the output--as many rows as there are eras or "program lines". The contents of the array are also added to any totaling buffers which are active (i.e. to as many rows of the /fltotl/ block as there have been BTOT lines given). Then the program rows are formatted and sent to the output buffer by calls to fladpg.

When an ETOT or EITOT keyword is encountered the "topmost" total row is sent to the output buffer and the number of active totals is decremented by one.

The JOB keyword line is unusual. Its purpose is to allow ships temporarily in out-of-force-repair-job status to appear on the report, so that they may be totaled. In this way an accurate representation of the number of ships actually in existence may be given in addition to an accurate representation of the number deployable. JOB line processing is undertaken by the fljob routine, which takes the given class list and searches the rejodat.(histj currj projj) relations for instances of jobs of interest on ships in the classes. Any found were certainly removed from the deployable force totals by the logic in ffltbl, so no double counting can result.

When flbrpt has completed processing of the format control file all work is essentially done. The flprnt routine rewinds the sequential disk file to which all output has been sent and writes its contents to the output device. Flclos then closes all files and relations and program processing terminates with a STOP. This automatically reactivates the System Core process and the user is returned to the Force Impact choice menu.

11.5.2 BGRP

An alphabetical list of the routines in program BGRP is given in Table 11-4. A calling tree diagram for BGRP is given in Figure 11-9. The logic of this program is very similar to that of FLRP up to the point of actual construction of the report. Note that ffltbl is used in both cases to conduct the data base search.

Bgbrpt (Battle Group Build RePoRt) faces a much different task than does flbrpt, however. Instead of summarizing the number of ships deployable in fairly raw terms, this routine must allocate scarce resource (the ships) among competing demands (the battle groups).

It does this on a period-by-period basis (i.e. its outer loop is over periods)---the allocation in any one period is independent of that in any other. Within a period, ships are allocated to the highest-priority battle group until its target number is reached, or until a constraint makes it impossible to have more of that particular group.

The requirements of each group are specified in terms of (possibly) broad functions, each of which may be filled (in descending order of preference) by several ship types, each of which in turn may be composed of several classes.

The type, therefore, is the lowest common denominator for purposes of the allocation. The number of ships of each type available in each period is computed from the contents of /fltabls/ and placed in the /typtot/ summary array before allocation begins.

The allocation is done on a trial-and-error basis, proceeding in order of priority and preference. For example, when computing the number of carrier battle groups as specified in the format control file in Figure 11-6, bgget (the allocation

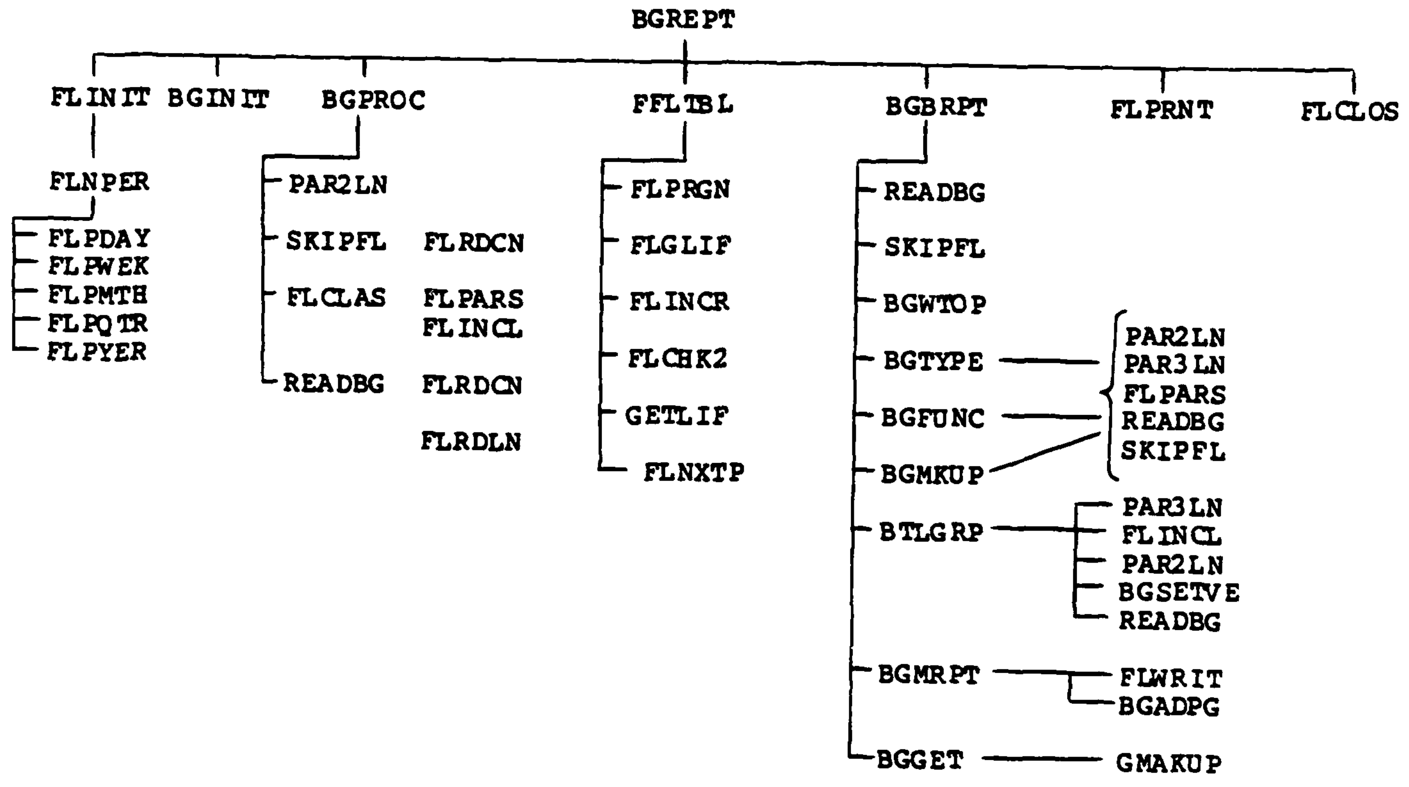
TABLE 11-4. Alphabetical Listing of Routines in BGRP Program

ROUTINE	HOST FILE
BGADPG	BGRPxxx
BGBRPT	BGRPxxx
BGFUNC	BGRPxxx
BGGET	BGRPxxx
BGINIT	BGRPxxx
BGMKUP	BGRPxxx
BGMRPT	BGRPxxx
BGPROC	BGRPxxx
BGREPT	BGRPxxx
BGSETV	BGRPxxx
BGTYPE	BGRPxxx
BGWRIT	BGRPxxx
BGWTOP	BGRPxxx
BTLGRP	BGRPxxx
FFLTBL	FLBGxxx
FLADPG	FLRPA
FLBUGI	FLBGxxx
FLCHK1	FLRPA
FLCHK2	FLBGxxx
FLCHK3	FLBGxxx
FLCLAS	FLBGxxx
FLCLOS	FLBGxxx
FLDECR	FLRPA
FLGLIF	FLBGxxx
FLINCL	FLBGxxx
FLINCR	FLBGxxx
FLINIT	FLRPA
FLJOB	FLBGxxx
FLNPER	FLBGxxx
FLNXTP	FLBGxxx
FLPARS	FLBGxxx
FLPDAY	FLBGxxx
FLPMTH	FLBGxxx
FLPQTR	FLBGxxx
FLPRGN	FLBGxxx
FLPRNT	FLBGxxx
FLPWEK	FLBGxxx
FLPYER	FLBGxxx
FLRDCN	FLBGxxx
FLRDLN	FLBGxxx
FLRPAGN	FLBGxxx
FLWRIT	FLBGxxx
FLWTOP	FLRPA
FNDPRD	FLBGxxx
GETJOB	FLBGxxx

TABLE 11-4. Alphabetical Listing of Routines in BGRP Program

ROUTINE	HOST FILE
GETLIF	FLBGxxx
GMAKUP	BGRPxxx
PAR2LN	FLBGxxx
PAR3LN	FLBGxxx
READBG	BGRPxxx
READFL	FLRPA
SKIPFL	FLBGxxx

Figure 11-9. BGRP Calling Tree Diagram



executive) would start by making up one battle group. It would do this by decrementing 1 carrier from function carrier, i.e. from the number of ships of type carrier available; then it would decrement 1 cruiser from function/type cruiser. If there were no ships left of type cruiser, it would try type BB, since that is an alternative for the cruiser function.

The decrementing that is taking place is being done on a temporary copy of the type availability array, so that if construction of a given group cannot be completed no "backing out" must be done to restore the actual count available.

Once all computations are complete the report is written out and flprnt and flclos are called to close files and clean up.

11.6 FILES USED BY THE FORCE IMPACT MODULE

Source code for FLRP alone is in flrpa.src. That used only by BGRP is in bgrpa.src, bgrpbgi.src, and bgrpbgw.src. Code for routines used jointly is in flbga.src, flbgflg.src, flbgflp.src, and flbgflr.src. Object code is in the complementary files in the .obj group. Combined (PREPable) object code is in flrp.obj and bgrp.obj. Program files are tflrp.prog, flrp.prog, tbgrp.prog, and bgrp.prog (development and production versions).

The format control input files are conventionally stored in the .fmtfil group.

Default output file when the report is saved to disk is flrept in the log-on group for both BGRP and FLRP.

Relations used are lccref.mnurel, vlrjob.mnurel, ncjodat.projj, ncjodat.currj, ncjodat.histj, rejodat.projj, rejodat.currj, rejodat.miscj, deact.miscj, and shlife.miscj.

11.7 SUMMARY OF INTERFACES

The Force Impact Module is fairly independent of other system components. It does use the standard Core services (scenario system, DBIF, and swap of Core data via a call to iniprc). The module is very dependent on the structure of the data base. Any change to the file or indexing structures of the relations listed in the previous section will be likely to render the module inoperative.

11.8 SUBROUTINE ABSTRACTS

Abstracts for both program FLRP and program BGRP are given on the following pages in alphabetical order. See Table 11-4 for a summary of the routines.

```
C     BGREPT***********************************************
$CONTROL check=3,segment=BGRP
      PROGRAM BGREPT
C*                     *** FORMAL PARAMETER DECLARATIONS ***
C*                                          *** ABSTRACT ***
C#PURPOSE exec for battle group force level report generator
C#AUDIT HISTORY
C          MEMutchler      27-JUN-83  AUTHOR
C#TYPE    main program
C#COMMON BLOCKS   none
C#CALLED BY menu system choice menu
C#METHOD
C  Initialize and open necessary relations and files.
C  Parse output control file creating an alphabetized list of
C  all ship classes found on type lines.  Fill in force level table,
C  one row for each ship class found, one column for each time period,
C  as number of ships built of that class in that time period - number
C  of ships of that class and period out for major deactivaing jobs or
C  in temporary retirement.  Process output control file along with
C  the force level table to build battle group force level report file.
C  Print force level report file.
C#LOCAL VARIABLES
C  err    error flag
C##
```

```
C     BGADPG*************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE BGADPG(TOTARRY,LAB)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE FLPMTR
%INCLUDE FLPERD
%INCLUDE FLHEAD
      CHARACTER LAB*LENRLB
      INTEGER TOTARRY(MXPERD)
C*                                          *** ABSTRACT ***
C#PURPOSE write total line to pagebuf
C#AUDIT HISTORY
C         MEMutchler      28-may-83  AUTHOR
C#TYPE    force level io routine
C#FORMAL PARAMETERS   none
C#COMMON BLOCKS
Cio       flpage  holds pagebuf
Cin       flhead  holds output text specs
Cin       fltotl  holds line to be output
Cin       flperd  holds period info
C#METHOD
C  write text to buf keeping track of lines used
C##
```

```
C     BGBRPT*************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE BGBRPT (ERR)
C*                          *** FORMAL PARAMETER DECLARATIONS ***
      LOGICAL ERR
C*                                                 *** ABSTRACT ***
C#PURPOSE use output control file and fltabl to create
C          force level report file
C#AUDIT HISTORY
C          MEMutchler      23-may-83  AUTHOR
C#TYPE     force level report utility
C#FORMAL PARAMETERS
Cin        err      error flag
C#COMMON BLOCKS
Cin        incpar  global parameters
Cin        charcon output control file keywords
Cin        readc   holds line number last read
Cio        fltotl  holds otaling arrays
Cin        fltabls holds force level tables for each program
C#CALLER   flreport
C#METHOD
C  process output control file by line creating the structure
C  for a force level report use values in fltabls for data.
C  Determine which type of line just read from output
C  control file and process accordingly
C    START  read everyline between a START and STOP line
C           only title and program labels are acknowledged
C           prior to the initil start
C    +      this is a continuation of the text of last line, only
C           for JOB or TYPE lines
C    TITLE  center text on the top of each report page
C           must be read before inital start
C    TYPE   text=typename,classes to make up type; defines a type
C           get values from force level tables for program types
C           and add class levels togeather
C    JOB    text=typename,jobname,job as known to RELATE,classes
C           in type; total all job done to each class for programs
C#LOCAL VARIABLES
C          protot  program totaling arrays
C##
```

```
C      BGFUNC**************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE BGFUNC(INFILE,IKEY,LINE,LENLINE,ERR,EOF)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE FLPMTR
      INTEGER INFILE,IKEY,LENLINE
      LOGICAL EOF,ERR
      CHARACTER LINE*LLINE
C*                                              *** ABSTRACT ***
C#PURPOSE process a fuction line from infile to get battle group
C#HISTORY
C        MEMutchler       16-may-83  AUTHOR
C#TYPE    process force level output control file
C#FORMAL PARAMETERS
Cin       infile  read from this file
Cio       ikey,line,lenline,err,eof   results of readfl
C#COMMON BLOCKS
Cio       readc   holds line counter
C#CALLER  flbldr
C#METHOD
C  get label off line, split rest of line into type names
C  assume first type found has top priority,etc.
C  process type names by storing index intotype totals as
C  fdefine(priority of choice,function it will achieve)
C  , process next line untill it is not a
C  continuation line
C##
```

```
C     BGGET*****************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE BGGET(ERR)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
C*                                               *** ABSTRACT ***
C#PURPOSE Compose the battle groups from the available ship
C         pool.
C#AUDIT HISTORY
C         MEMutchler      29-JUB-83  AUTHOR
C#TYPE    battle group counting
C#FORMAL PARAMETERS
Cout      err     true if major error was found
C#COMMON BLOCKS
Cin      groupbg  descrip of battle group compositions, tgts
Cin      typebg   ships available by type
Cin      funcbg   functional ship family definitions
C#CALLER  bgbrpt
C#METHOD
C         given data from forcelevel table and battlegroup
C         output control, make up battle group force, one
C         period at a time, filling first priority groups
C         first, with first choice type to do a function
C#LOCAL VARIABLES
C        temptypt buffer of per-period ships avail for a type
C        grprior  group priority buffer
C##
```

```
C     BGINIT*****************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE BGINIT
C*                    *** FORMAL PARAMETER DECLARATIONS ***
C*                                         *** ABSTRACT ***
C#PURPOSE initialize BGRP internal buffers.
C         enddate
C#AUDIT HISTORY
C         MEMutchler     31-may-83  AUTHOR
C#TYPE    intialize arrays to zero
C#FORMAL PARAMETERS
C#COMMON BLOCKS
Cout     typebg   ship type avail info
Cout     groupbg  battle groups defns
Cout     funcbg   functional families
C#METHOD
C        Loops setting array locations to zero.
C##
```

```
C     BGMKUP***************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE BGMKUP(INFILE,IKEY,LINE,LENLINE,ERR,EOF)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE FLPMTR
      INTEGER INFILE,IKEY,LENLINE
      LOGICAL EOF,ERR
      CHARACTER LINE*LLINE
C*                                          *** ABSTRACT ***
C#PURPOSE process a makeup line from infile to get forcelevl
C#HISTORY
C        MEMutchler       27-JUN-83  AUTHOR
C#TYPE    process battlegroup force level output control file
C#FORMAL PARAMETERS
Cin       infile  read from this file
Cio       ikey,line,lenline,err,eof   results of readfl
C#COMMON BLOCKS
Cio       readc   holds line counter
Cin       flclass holds class list
C#CALLER  flbldr
C#METHOD
C  get label off line, split rest of line into function names
C  ,number of function members needed for this battlegroup
C  When done with all
C  functions from line, process next line untill it is not a
C  continuation line
C##
```

```
C     BGMRPT**************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE BGMRPT
C*                        *** FORMAL PARAMETER DECLARATIONS ***
C*                                               *** ABSTRACT ***
C#PURPOSE writes battle group report to ioutfl
C#AUDIT HISTORY
C          MEMutchler      28-may-83  AUTHOR
C#TYPE     fore level io routine
C#FORMAL PARAMETERS   none
C#COMMON BLOCKS
Cio        flpage  holds pagebuf
Cin        flhead  holds header text
C#METHOD
C  write text to buf keeping track of lines used
C##
```

```
C     BGPROC***************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE BGPROC(ERROR)
C*                          *** FORMAL PARAMETER DECLARATIONS
      LOGICAL ERROR
C*                                                   *** ABSTRACT ***
C#PURPOSE parse output control file and create an alphabetized
C          list of each ship class mentioned on a TYPE line.
C#AUDIT HISTORY
C
C          MEMutchler      27-JUN-83  AUTHOR
C#TYPE     find which classes are to be examined
C#FORMAL PARAMETERS      non
C#COMMON BLOCKS
Cin        ioc     i/o file assignments
Cin        incpar  global parameters
C#CALLER   flreport
C#METHOD  Starting at top of FILE OCNTRL look at a  line
C  IF line begins wih "TYPE" THEN extract class names from line and
C  following lines begining with "+", add the names to a
C  list of names IF not already there.  ELSE GO TO next line untill
C  end of file.  Alphabetize the list of names.
C#LOCAL VARIABLES
C          line    one line from ocfile
C          lenline deblanked length of line
C          eof     true IFf end of file ocfile has been read
C          lenkey  deblanked length of key
C##
```

```
C     BGSETV**************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE BGSETV(BEGDATE,ENDDATE,PROGRAM,VALUE)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE FLPMTR
%INCLUDE FLPERD
      INTEGER*4 BEGDATE,ENDDATE
      LOGICAL DEARLY
      INTEGER VALUE, PROGRAM (1,MXPERD)
C*                                          *** ABSTRACT ***
C#PURPOSE increment program total for period rom begdate to
C         enddate
C#AUDIT HISTORY
C         MEMutchler      31-may-83  AUTHOR
C#TYPE    set value due to dates
C#FORMAL PARAMETERS
Cin          begdate begining date
Cin       enddate ending date
Cio       program holds current program totals
C#COMMON BLOCKS
Cin      flperd   time horizon this run
C#METHOD
C  get first period after begindate, get last period before
C  enddate.  Increment program between these two periods
C##
```

```
C      BGTYPE**************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE BGTYPE(INFILE,IKEY,LINE,LENLINE,ERR,EOF)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE FLPMTR
      INTEGER INFILE,IKEY,LENLINE
      LOGICAL EOF,ERR
      CHARACTER LINE*LLINE
C*                                           *** ABSTRACT ***
C#PURPOSE process a type line from infile to get forcelevl
C#HISTORY
C        MEMutchler       16-may-83  AUTHOR
C#TYPE    process force level output control file
C#FORMAL PARAMETERS
Cin       infile  read from this file
Cio       ikey,line,lenline,err,eof   results of readfl
C#COMMON BLOCKS
Cio       readc   holds line counter
Cin       flclass holds class list
C#CALLER  flbldr
C#METHOD
C  get label off line, split rest of line into class names
C  process class names by adding force levels of eachclass
C  mentioned to the aprropriate prgbuf.  When done with all
C  classes from line, process next line untill it is not a
C  continuation line
C##
```

```
C     BGWRIT***************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE BGWRIT
C*                      *** FORMAL PARAMETER DECLARATIONS ***
C*                                                *** ABSTRACT ***
C#PURPOSE write pagebuf to ioutfl
C#AUDIT HISTORY
C         MEMutchler      28-may-83  AUTHOR
C#TYPE    battle group level io routine
C#FORMAL PARAMETERS   none
C#COMMON BLOCKS
Cio       flage  holds pagebuf
Cin       flhead  holds output text specs
C#METHOD
C  write text to buf keeping track of lines used
C##
```

```
C     BGWTOP************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE BGWTOP
C*                      *** FORMAL PARAMETER DECLARATIONS ***
C*                                          *** ABSTRACT ***
C#PURPOSE writes title line and period header to pagebuf
C#AUDIT HISTORY
C         MEMutchler      28-may-83  AUTHOR
C#TYPE    fore level io routine
C#FORMAL PARAMETERS   none
C#COMMON BLOCKS
Cio       flpage  holds pagebuf
Cio       flhead  header text
C#METHOD
C  write text to buf keeping track of lines used
C#LOCAL VARIABLES
C        none
C##
```

```
C      BTLGRP***************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE BTLGRP(INFILE,IKEY,LINE,LENLINE,ERR,EOF)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE FLPMTR
      INTEGER INFILE,IKEY,LENLINE
      LOGICAL EOF,ERR
      CHARACTER LINE*LLINE
C*                                         *** ABSTRACT ***
C#PURPOSE process a bgroup line from infile to get forcelevl
C#HISTORY
C        MEMutchler       27-JUN-83  AUTHOR
C#TYPE    process battlegroup force level output control file
C#FORMAL PARAMETERS
Cin       infile  read from this file
Cio       ikey,line,lenline,err,eof   results of readfl
C#COMMON BLOCKS
Cio       readc   holds line counter
Cin       flclass holds class list
C#CALLER  flbldr
C#METHOD
C  get groupname off line, split rest of line into group label(to
C  be printed on actual output),priority with which this group must
C  be filled, focre level to achive, date this info begins at,
C  date this info ends wtih, process next line untill it is not a
C  continuation line
C##
```

```
C     FFLTBL(ERR)*******************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FFLTBL(ERR)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      LOGICAL ERR
C*                                          *** ABSTRACT ***
C#PURPOSE fill in the force level table (fltabl) buffer
C     giving ships available each period
C#AUDIT HISTORY
C          MEMutchler/MCarey      23-may-83  AUTHOR
C#FORMAL PARAMETERS
C         none
C#COMMON BLOCKS
Cio       readc    number of last input file line, for lwarn
Cin       flcons   file and field names for relation access
Cin       fldecm   "
Cin       flrjob   "
Cin       shlife   "
Cin       scenar   current scenario info
Cout      fltabls  output table, ships by period and program
C#CALLER  FLREPORT
C#METHOD
C  for each of the alphabetized ship classes
C  get the standard service lIFe of any in the class
C    for each hull # in the class
C    search  the construction relations for first construction
C    search for first decommisionings
C    search for all other construction and decommisionings
C    search job relation for all jobs
C    keep job info IF it adds to service lIFe or in major job list
C
C#LOCAL VARIABLES
C          endlIF  true IFf no decommision date found for
C                  a recomissioning
C          maxcons max. # of contructions done to one ship
C          maxjob  max # of repair jobs done to one ship
C          jblIFe  years added to lIFe due to this job
C          cnlIFe  years added to lIFe due to this construction
C          condat  date of construction delivery
C         decdat  date of decommisioning
C          jbdat   date of begining  of repair job
C          jedat   date of ending of repair job
C          invbuf  inventory buffer
C          prgbuf  program buffer
C          majjob  true IF job i is one of lsiton
C          conpro  construction is program not inventory
C          jobpro  repair job is program not inventory
C##
```

```
C     FLADPG**************************************************
$CONTROL check=3,segment=FLRP
      SUBROUTINE FLADPG(TOTARRY,LLAB,RLAB,TOTALIN)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE FLPMTR
%INCLUDE FLPERD
%INCLUDE FLHEAD
      CHARACTER LLAB*LENLLB, RLAB*LENRLB
      INTEGER TOTARRY(MXPERD)
       LOGICAL TOTALIN
C*                                           *** ABSTRACT ***
C#PURPOSE write total line to pagebuf
C#AUDIT HISTORY
C          MEMutchler      28-may-83  AUTHOR
C#TYPE    force level io routine
C#FORMAL PARAMETERS   none
C#COMMON BLOCKS
Cio       flpage  holds pagebuf
Cin       flhead  holds output text specs
Cin       fltotl  holds line to be output
Cin       flperd  holds period info
C#METHOD
C  write text to buffer keeping track of lines used
C##
```

```
C     FLBRPT***************************************************
$CONTROL check=3,segment=FLRP
      SUBROUTINE FLBRPT (ERR)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      LOGICAL ERR
C*                                          *** ABSTRACT ***
C#PURPOSE use output control file and fltabl to create
C         force level report file
C#AUDIT HISTORY
C         MEMutchler      23-may-83  AUTHOR
C#TYPE    force level report utility
C#FORMAL PARAMETERS
Cin       err      error flag
C#COMMON BLOCKS
Cin       incpar  global parameters
Cin       charcon output control file keywords
Cin       readc   holds line number last read
Cio       fltotl  holds otaling arrays
Cin       fltabls holds force level tables for each program
C#CALLER  flreport
C#METHOD
C  process output control file by line creating the structure
C  for a force level report use values in fltabls for data.
C     Determine which type of line just read from output
C  control file and process accordingly
C    START  read everyline between a START and STOP line
C           only title and program labels are acknowledged
C           prior to the initil start
C    +      this is a continuation of the text of last line, only
C           for JOB or TYPE lines
C    TITLE  center text on the top of each report page
C           must be read before inital start
C    PROGLB text is the progra label,program start date
C           at least one must be read before start
C    BTOT   begin a new total array of beginname=text
C           add a new totaling array
C    ETOT   end the last total array begun,text=name,left
C           label,ight label, be sure name=beginname
C           write report lines to report file
C           delete last totaling array
C    EITOT  end the last total array begun,text=name,left
C         label,right label, be sure name=beginname
C           don't o a page feed  after, delete totaling array
C    TYPE   text=typename,classes to make up type; defines a type
C           get values from force level tables for program types
C           and add class levels togeather
C    JOB    text=typename,jobname,job as known to RELATE,classes
C           in type; total all job done to each class for programs
C#LOCAL VARIABLES
C         protot  program totaling arrays
C##
```

```
C     FLCHK1**************************************************
$CONTROL check=3,segment=FLRP
      LOGICAL FUNCTION FLCHK1(SCEN1,SCEN2,CLAS1,CLAS2,INT1,INT2)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*12 SCEN1,SCEN2
      CHARACTER*10 CLAS1,CLAS2
      INTEGER INT1,INT2
C*                                                 *** ABSTRACT ***
C#PURPOSE make sure all *1=*2
C#AUDIT HISTORY
C         MEMutchler      31 may 83  AUTHOR
C#TYPE    force level utility
C#FORMAL PARAMETERS
Cin     scen     scenario name
Cin     clas     class name
Cin     int      number
Cou     flchk1   true if all *1 = *2
C  check each pair
C##
```

```
C     FLCHK2*************************************************
$CONTROL check=3,segment=FLB6
      LOGICAL FUNCTION FLCHK2(SCEN1,SCEN2,CLAS1,CLAS2,
     +     INTA1,INTA2,INTB1,INTB2)
C*                     *** FORMAL PARAMTER DECLARATIONS ***
      CHARACTER*12 SCEN1,SCEN2
      CHARACTER*10 CLAS1,CLAS2
      INTEGER INTA1,INTA2,INTB1,INTB2
C*                                          *** ABSTRACT ***
C#PURPOSE make sure all *1=*2; used for end-of-data detection
C     when reading along a RELATE index
C# HISTORY
C        MEMutchler       31 may 83  AUTHOR
C#TYPE    force level utility
C#FORMAL PARAMETERS
Cin     scen     scenario name
Cin     clas     class name
Cin     int      number
Cou     flchk2   true if all *1 = *2
C#METHOD
C  check each pair
C##
```

```
C     FLCHK3**********************************************************
$CONTROL check=3,segment=FLBG
      LOGICAL FUNCTION FLCHK3(SCEN1,SCEN2,CLAS1,CLAS2,
     +     INTA1,INTA2,JOBID1,JOBID2)
C*                       *** FORMAL PARAMTER DECLARATIONS ***
      CHARACTER*12 SCEN1,SCEN2
      CHARACTER*10 CLAS1,CLAS2
      CHARACTER*8 JOBID1,JOBID2
      INTEGER INTA1,INTA2
C*                                            *** ABSTRACT ***
C#PURPOSE make sure all *1=*2; for RELATE end-of-data-group
C     detection.
C# HISTORY
C        MEMutchler       31 may 83  AUTHOR
C#TYPE    force level utility
C#FORMAL PARAMETERS
Cin     scen      scenario name
Cin     clas      class name
Cin     int       number
Cou     flchk2    true if all *1 = *2
C#METHOD
C  check each pair
C##
```

```
C     FLCLAS ************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLCLAS( LINE,LENLINE,CONTLN,ERR)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER LENLINE
      LOGICAL ERR
      CHARACTER LINE*(LENLINE)
      LOGICAL CONTLN
C*                                                   *** ABSTRACT ***
C#PURPOSE parse the text part of a type line from the format
C         file, adding class names to the list of class names
C         IF not yet present
C#AUDIT HISTORY
C         MEMutchler      17-may-83  AUTHOR
C#TYPE    parse output control file
C#FORMAL PARAMETERS
Cin       line     the text following the keyword in the output control
C                  file
Cin       lenline  length of theline
Cin       contln   true IF this was a continue line,else false
C#COMMON BLOCKS
Cio       classes     holds class list
Cin       incpar      global parameter list
Cin       charcon     constant strings
Cin       readc       holds line number just read
C#CALLER  parsoc
C#METHOD  remove first phrase if not contln.  Parse line, one
C  phrase at a time, adding to list if possible.
C#LOCAL VARIABLES
C         string  unparsed prt of line
C         lstring deblanked length of string
C         clname  one class name
C##
```

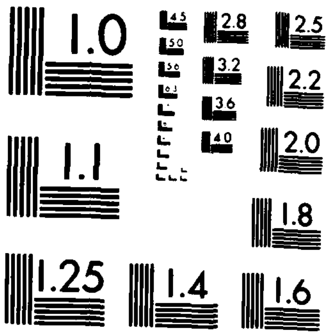
```
C     FLCLOS*************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLCLOS
C*                        *** FORMAL PARAMETER DECLARATIONS ***
C*                                            *** ABSTRACT ***
C#PURPOSE close files and relations only needed by flrept
C#AUDIT HISTORY
C          MEMutchler      19-may-83  AUTHO
C#TYPE     clean up
C#FORMAL PARAMETERS
C         none
C#COMMON BLOCKS
Cin       flioc    fortran io units
Cin       fldecm   relation names
Cin       flrjob   "
Cin       shlife   "
C#CALLER   flrept
C#METHOD
C  Calls to filcls and rvclos; also call lpsend to start
C  printing.
C##
```

```
C     FLDECR*************************************************
$CONTROL check=3,segment=FLRP
      SUBROUTINE FLDECR(BEGDATE,ENDDATE,PROGRAM)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE FLPMTR
%INCLUDE FLPERD
      INTEGER*4 BEGDATE,ENDDATE
      LOGICAL DEARLY
      INTEGER PROGRAM (1,MXPERD)
C*                                          *** ABSTRACT ***
C#PURPOSE decrement program total for period from begdate to
C         enddate
C#AUDIT HISTORY
C         MEMutchler      31-may-83  AUTHOR
C#TYPE    decrement due to dates
C#FORMAL PARAMETERS
Cin       begdate repair job begining date
Cin       enddate repair job ending date
Cio       program holds current program totals
C#COMMON BLOCKS
Cin      flperd   first day of each period
C#METHOD
C  get first period after begindate, get last period before
C  enddate.  Decrement program between these two priods
C##
```

```
C     FLGLIF ************************************************
$CONTROL segment=flbg
      SUBROUTINE flglif(stanlf,stunit,ncom,condat,decdat,
     1                  addlif,addunt,njob,joblif,jobunt,
     2                  jedate,err)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      integer ncom,njob,stanlf,addlif(ncom),joblif(njob)
      integer*4 condat(ncom),decdat(ncom),jedate(njob)
      character*6 stunit,addunt(ncom),jobunt(njob)
      logical err
C*                                          *** ABSTRACT ***
C#PURPOSE   get last decommissioning date for ship
C#AUDIT HISTORY
C         MEMutchler     31-may-83  AUTHOR
C         MSCarey        28-apr-84  Major logic change to use
C                                   time units in calculations
C#FORMAL PARAMETERS
Cin       stanlf   standard life of ship
Cin       stunit   in these units
Cin       ncom     number of commissionings ship has
Cin       condat   commissioning dates
Cio       decdat   deactivation dates; output in decat(ncom) <---**
Cin       addlif   amount of life added each commissioning
Cin       addunt   in these time units
Cin       njob     number of repair jobs ship has had
Cin       joblif   amount of life added by each repair job
Cin       jobunt   in these time units
Cin       jedate   end date of each repair job
Cout      err      true if any jedate>decdat(ncom)
C#COMMON BLOCKS
C         none
C#CALLER ffltbl
C#METHOD
C     Obtain the number of days used during each commissioning.
C     Obtain the retirement date which would have occurred if
C     there was only one commissioning, and if there was no
C     life added by any job, using the standard lifetime/units.
C     Find the number of days represented by this lifetime.
C     Find the number of days between the first activation and
C     the last activation, and compare this to the number
C     of days used during active periods to get the amount of
C     time spent in mothballs.  Get a final deactivation date.
C     Then cycle through the repair
C     jobs and push out this date by the amount of time added
C     by each job.  Then cycle through the commissioning jobs
C     and do the same.  The result is a computed deactivation
C     date based on (possibly) different time units used to
C     to specify the various life-length increments.
C#LOCAL VARIABLES
C         comusd   days of life used up by commissioned time
C         mthadd   days of life 'added' by time spent in mothballs
C         rawday   length of standard life in days
C         rawdat   raw retirement date
C         deact    working retirement date
C##
```

AD-A150 423 ALIAS (ACQUISITION AND LOGISTICS INFORMATION AND ANALYSIS SYSTEM) MAINTEN..(U) DECISION-SCIENCE APPLICATIONS INC ARLINGTON VA M S CAREY ET AL. 5/7

UNCLASSIFIED 31 OCT 84 DSA-593-VOL-2 N00014-82-C-0813 F/G 15/5 NL


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

```
C     FLINCL*************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLINCL(STR,LSTR,LSIZE,MLSIZE,LIST,ELSIZE,
     1 IINDEX,ERR)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER STR*(ELSIZE) ,LIST*(ELSIZE)(MLSIZE)
      LOGICAL ERR
      INTEGER IINDEX,LSTR,LSIZE,MLSIZE,ELSIZE
C*                                              *** ABSTRACT ***
C#PURPOSE include string in a list
C#AUDIT HISTORY
C         MEMutchler      8  JUN 83  AUTHOR
C#TYPE    force level report generator utility
C#FORMAL PARAMETERS
CIN   STR      NAME TO GO ON LIST
CIN   LSTR     NUMBER OF CHARS IN STR
CIO   LSIZE    NAME-LIST SIZE.  WILL BE INCREMENTED IF STR NOT
C              ALREADY ON THE LIST.
CIN   MLSIZE   MAX ALLOWED VALUE OF LSIZE
CIO   LIST     NAME-LIST
CIN   ELSIZE   MAX CHARS IN EACH ELEMENT OF 'LIST'
COUT  IINDEX   INDEX OF STR ON LIST
C#COMMON BLOCKS
Cin       readc   # of line last read from input file
C#METHOD
C  CONSTRUCTS LIST 'LIST' OF NAMES
C##
```

```
C      FLINCR*************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLINCR(BEGDATE,ENDDATE,PROGRAM)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE FLPMTR
%INCLUDE FLPERD
      INTEGER*4 BEGDATE,ENDDATE
      LOGICAL DEARLY
      INTEGER PROGRAM (1,MXPERD)
C*                                          *** ABSTRACT ***
C#PURPOSE increment program total for period from begdate to
C         enddate
C#AUDIT HISTORY
C         MEMutchler      31-may-83  AUTHOR
C#TYPE    increment due to dates
C#FORMAL PARAMETERS
Cin          begdate begining date
Cin       enddate ending date
Cio       program holds current program totals
C#COMMON BLOCKS
Cin      flperd   .first day of each period
C#METHOD
C  get first period after begindate, get last period before
C  enddate.  Increment program between these two periods
C##
```

```
C     FLJOB***************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLJOB(INFILE,IKEY,LINE,LENLINE,ERR,EOF)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE FLPMTR
      INTEGER INFILE,IKEY,LENLINE
      LOGICAL EOF,ERR
      CHARACTER LINE*LLINE
C*                                           *** ABSTRACT ***
C#PURPOSE get the total effect on classes seen on one job
C line of the output control file as performed by periods
C#AUDIT HISTORY
C         MEMutchler      31-may-83  AUTHOR
C#TYPE    fill total array by processing ocfile
C#FORMAL PARAMETERS
Cin       infile  read from this file
Cio       ikey,line,lenline,err,eof   results f readfl
C#COMMON BLOCKS
Cio       readc   holds line counter
C#CALLER  flbldr
C#METHOD
C  get label and jobname off line split rest of line into classes
C  process the job by totalin for each period how many times
C  that job was performed on any of the classess mentioned
C  When done processing all of the
C  classes from line, process next line untill it is not a
C  continuation line
C##
```

```
C     FLNPER************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLNPER(ERROR)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
      LOGICAL ERROR
C*                                                 *** ABSTRACT ***
C#PURPOS determine nperiod and fill in datper(1..nperiod)
C          and makes up period header
C          MEMutchler      31-may-83  AUTHOR
C#TYPE    report utility
C#COMMON BLOCKS
Cin        value  menu parameter values
Cin        pvdecl  menu parameter declarations
Cin        pveqiv  menu parameter equivalences
Cou        flperd  period info
C#CALLER  FLINIT
C#METHOD
C  Calls to a subsidiary routines, depending on period length
C  Note--implemented before standard TDDATE ALIAS date utilities
C##
```

```
C     FLNXTP*********************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLNXTP(ANYDAT,PERDAT)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER*4 ANYDAT,PERDAT
      LOGICAL DEARLY
C*                                           *** ABSTRACT ***
C#PURPOSE get first datper followng anydat
C#AUDIT HISTORY
C         MEMutchler      2-june-83  AUTHOR
C#TYPE    force level date utility
C#FORMAL PARAMETERS
Cin       anydat  relate clarified *4 date
Cou       perdat  first datein datper following anydat
C#COMMON BLOCKS
Cin       flperd  holds period info
C#CALLER  flglif
C#METHOD
C  loop through datper array untill a date .ge. anydat is found
C#LOCAL VARIALES
C         iperd   period index
C##
```

```
C     FLPARS ***********************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE flpars(line,lenlin,list,lenlst,mxlchr,mxnum,num,
     1                  tomany,tolong)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      character*(lenlin) line
      character*(mxlchr) list(mxnum)
      integer lenlin,mxnum,num,lenlst(mxnum)
      logical tomany,tolong
C*                                            *** ABSTRACT ***
C#PURPOSE   Parses an input string into substrings delimited
C           by commas.
C#AUDIT HISTORY
C       MSCarey         03-jun-83  AUTOR
C#FORMAL PARAMETERS
Cin     line     string to be parsed
Cin     lenlin   length of line in chars
Cout    list     list of output substrings
Cin     mxlchr   max length of any substring
Cout    lenlst   length of each substring
Cin     mxnum    maximum number of substrings returnable
Cout    num      number of substrings found
Cout    tomany   true if more than mxnum substrings found
Cout    tolong   true if a substring longer than mxlchr found
C#COMMON BLOCKS
C       none
C#CALLER various
C#METHOD
C     Look for commas and extract the intermediate text.
C##
```

```
C     FLPDAY*************************************************
$CONTROL check=3,segment=FLB6
      SUBROUTINE FLPDAY(ERROR)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
      LOGICAL ERROR
C*                                             *** ABSTRACT ***
C#PURPOSE determine nperiod and fill in datper(1..nperiod)
C        and makes up period header when period length=day
C#AUDIT HISTORY
C        MEMutchler      31-may-83  AUTHOR
C#TYPE   report utility
C#COMMON BLOCKS
Cin      pvalue  menu parameter values
Cin      pvdecl  menu parameter declarations
Cin      pveqiv  menu parameter equivalences
Cou      flperd  period info
C#CALLER FLINIT
C#METHOD
C  Low-level date utility calls and straight string concats.
C##
```

```
C     FLPMTH**************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLPMTH(ERROR)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      LOGICAL ERROR
C*                                             *** ABSTRACT ***
C#PURPOSE determine nperiod and fill in datper(1..nperiod)
C         and makes up period header when period length=year
C#AUDIT HISTORY
C         MEMutchler      31-may-83  AUTHOR
C#TYPE    report utility
C#COMMON BLOCKS
Cin       pvalue  menu parameter values
Cin       pvdecl  menu parameter declarations
Cin       pveqiv  menu parameter equivalences
Cou       flperd  period info
C#CALLER  FLINIT
C#METHOD
C  Low-level date utility calls and straight string concats.
C##
```

```
C     FLPQTR***************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLPQTR(ERROR)
C*                     ** FORMAL PARAMETER DECLARATIONS ***
      LOGICAL ERROR
C*                                          *** ABSTRACT ***
C#PURPOSE determine nperiod and fill in datper(1..nperiod)
C         and makes up period header when period length=year
C#AUDIT HISTORY
C         MEMutchler      31-may-83  AUTHOR
C#TYPE    report utility
C#COMMON BLOCKS
Cin       pvalue  menu parameter values
Cin       pvdecl  menu parameter declarations
Cin       pveqiv  menu parameter equivalnces
Cou       flperd  period info
C#CALLER  FLINIT
C#METHOD
C  Low-level date utility calls and straight string concats.
C##
```

```
C     FLPRGN*************************************************
$CONTROL check=3,segment=FLBG
      INTEGER FUNCTION FLPRGN(APPROP,AWARD,DELIV)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER*4 APPROP,AWARD,DELIV
C*                                              *** ABSTRACT ***
C#PURPOSE get program number this will fall in according
C         to appropriate date
C#AUDIT HISTORY
C         MEMutchler      31-may-83  AUTHOR
CTYPE     force level utility
C#FORMAL PARAMETERS
Cin       approp  appropriation date
Cin       award   award date
Cin       deliv   delivery date
C#COMMON BLOCKS
Cin       pvalue  menu parameter values
Cin       pvdecl  menu parameter declarations
Cin       pvequiv menu parameter equivalences
Cin       fltabls program begining dates
C#CALLER  ffltbl
C#METHOD  determine which date to use to determine program
C  and use it with begining program dates to find program
C  the date falls in.
C##
```

```
C     FLPRNT**************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLPRNT
C*                      *** FORMAL PARAMETER DECLARATIONS ***
C*                                            *** ABSTRACT ***
C#PURPOSE print report to daisy or lp
C#AUDIT HISTORY
C         MEMutchler      16-may-83  AUTHOR
C#FORMAL PARAMETERS
C        none
C#COMMON BLOCKS
Cin      flioc    flrp unit numbers
C#METHOD
C    The sequential file contains the actual force level report to
C    be displayed using SUBROUTINE FLPRNT.  It contains the actual
C    lines of text,titles,non-printing comments, and page feed markers.
C    It must be made permanent IF it is to be saved, and may be edited
C    IF desired.  Print all printable lines literally, and use page ejec
C#LOCAL VARIABLES
C         line    a line of text to be printed
C##
```

```
C     FLPROC*************************************************
$CONTROL check=3,segment=FLRP
      SUBROUTINE FLPROC(ERROR)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      LOGICAL ERROR
C*                                              *** ABSTRACT ***
C#PURPOSE parse output control file and create an alphabetized
C          list of each ship class mentioned on a TYPE line.
C#AUDIT HISTORY
C          MEMutchler      16-may-83  AUTHOR
C#TYPE     find which classes are to be examined
C#FORMAL PARAMETERS      non
C#COMMON BLOCKS
Cin        ioc     i/o file assignments
Cin        incpar  global parameters
C#CALLER  flreport
C#METHOD  Starting at top of FILE OCNTRL look at a  line
C  IF line begins wih "TYPE" THEN extract class names from line and
C  following lines begining with "+", add the names to a
C  list of names IF not already there.  ELSE GO TO next line untill
C  end of file.  Alphabetize the list of names.
C#LOCAL VARIABLES
C          line    one line from ocfile
C          lenline deblanked length of line
C          eof     true IFf end of file ocfile has been read
C          lenkey  deblanked length of key
C##
```

```
C     FLPWEK*****************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLPWEK(ERROR)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      LOGICAL ERROR
C*                                            *** ABSTRACT ***
C#PURPOSE determine nperiod and fill in datper(1..nperiod)
C         and makes up period header when period length=week
C#AUDIT HISTORY
C         MEMutchler      31-may-83  AUTHOR
C#TYPE    report utility
C#COMMON BLOCKS
Cin       pvalue  menu parameter values
Cin       pvdecl  menu parameter declarations
Cin       pveqiv  menu parameter equivalences
Cou       flperd  period info
C#CALLER  FLINIT
C#METHOD
C  Low-level date utility calls and straight string concats.
C##
```

```
C     FLPYER*************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLPYER(ERROR)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      LOGICAL ERROR
C*                                          *** ABSTRACT ***
C#PURPOSE determine nperiod and fill in datper(1..npeiod)
C         and makes up period header when period length=year
C#AUDIT HISTORY
C         MEMutchler     31-may-83  AUTHOR
C#TYPE    report utility
C#COMMON BLOCKS
Cin       pvdecl  menu parameter declarations
Cin       pveqiv  menu parameter equivalences
Cou       flperd  period info
C#CALLER  FLINIT
C#METHOD
C  Low-level date utility calls and straight string concats.
C##
```

```
C     FLRDCN*****************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLRDCN (IUNIT,LINE,EOF)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE LPRNTS
       LOGICAL EOF
       INTEGER IUNIT
       CHARACTER LINE*LLINE
C*                                               *** ABSTRACT ***
C#PURPOSE read from file IN and keep track of lines read
C#AUDIT HISTORY
C         MEMutchler          17 JAN 83  AUTHOR
C         MEMutchler          8  FEB 83 TESTER  (program treadc)
C#TYPE    mnugen utility
C#FORMAL PARAMETERS
Cin       iunit   file number from which to read
Cout      line    input line read
Cout      eof     true iff eof read from iunit
C#COMMON BLOCKS
Cin       incpar  global parameter statementa
Cin       reads   holds iline
C#METHOD. An unformated read is done from unit =
C         iunit.  EOF = false unless an end of file is read
C         in which case EOF = true.  If command file building
C         is in use,  LINE is echoed to unit = icomfile.
C         Icount is incremented.
C#LOCAL VARIABLES
C         recch   '%' recognition character for comment card
C##
```

```
C     FLRDLN***************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLRDLN (IUNIT,LINE,EOF)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE IOC
      INTEGER IUNIT
      LOGICAL EOF
      CHARACTER LINE*LLINE,BUFFER*LLINE
C*                                              *** ABSTRACT ***
C#PURPOSE read a line from IUNIT
C#AUDIT HISTORY
C         MEMutchler           10 JUN 83  AUTHOR
C#TYPE    flrept utility
C#FORMAL PARAMETERS
Cin       iunit   unit number from which to read
Cout      line    line that was read
Cout      eof     true iff eof was read
C#COMMON BLOCKS
Cin       incpar  global parameter statement
Cin       comcfl  holds command file irfo.
C#METHOD  An unformated read is done from unit =
C         iunit.  EOF = false unless an end of file is read
C         in which case EOF = true.  If command file building
C         is in use,  LINE is echoed to unit = icomfile.
C#LOCAL VARIABLES   none
C##
```

```
C     FLREPT***********************************************
$CONTROL check=3,segment=FLRP
      PROGRAM FLREPT
C*                      *** FORMAL PARAMETER DECLARATIONS ***
C*                                            *** ABSTRACT ***
C#PURPOSE main for force level report generator
C#AUDIT HISTORY
C          MEMutchler      16-MAY-83  AUTHOR
C#TYPE    main program
C#COMMON BLOCKS   none
C#CALLED BY menu system choice menu
C#METHOD
C  Initialize and open necessary relations and files.
C  Parse output control file creating an alphabetized list of
C  all ship classes found on type lines.  Fill in force level table,
C  one row for each ship class found, one column for each time period,
C  as number of ships built of that class in that time period - number
C  of ships of that class and period out for major deactivaing jobs or
C  in temporary retirement.  Process output control file along with
C  the force level table to build force level report file.
C  Print force level report file.
C#LOCAL VARIABLES
C  err     error flag
C##
```

```
C     FLRPGN*****************************************************
$CONTROL check=3,segment=FLBG
      INTEGER FUNCTION FLRPGN(DATE,NCONS,CONDATE)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER NCONS
      INTEGER*4 DATE, CONDATE(NCONS)
      LOGICAL DEARLY
C*                                              *** ABSTRACT ***
C#PURPOSE get program number this will fall in according
C         to date
C#AUDIT HISTORY
C         MEMutchler      31-may-83  AUTHOR
C#TYPE    force level utility
C#FORMAL PARAMETERS
Cin       date    repair begiing date
C#COMMON BLOCKS
Cn        pvalue  menu parameter values
Cin       pvdecl  menu parameter declarations
Cin       pvequiv menu parameter equivalences
Cin       fltabls program begining dates
C#CALLER  ffltbl
C#METHOD
C  use date with begining program dates to find program
C  the date falls in.
C##
```

Y c l

```
C     FLTYPE*************************************************
$CONTROL check=3,segment=FLRP
      SUBROUTINE FLTYPE(INFILE,IKEY,LINE,LENLINE,ERR,EOF)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE FLPMTR
      INTEGER INFILE,IKEY,LENLINE
      LOGICAL EOF,ERR
      CHARACTER LINE*LLINE
C*                                            *** ABSTRACT ***
C#PURPOSE process a type line from infile to get forcelevl
C#AUDIT HISTORY
C        MEMutchler      16-may-83  AUTHOR
C#TYPE    process force level output control file
C#FORMAL PARAMETERS
Cin       infile  read from this file
Cio       ikey,line,lenline,err,eof   results of readfl
C#COMMON BLOCKS
Cio       readc   holds line counter
Cin       flclass holds class list
C#CALLER  flbldr
C#METHOD
C  get label off line, split rest of line into class names
C  process class names by adding force levels of eachclass
C  mentioned to the aprropriate prgbuf.  When done with all
C  classes from line, process next line untill it is not a
C  continuation line
C##
```

```
C     FLWRIT*********************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FLWRIT
C*                       *** FORMAL PARAMETER DECLARATIONS ***
C*                                            *** ABSTRACT ***
C#PURPOSE write pagebuf to ioutfl
C#AUDIT HISTORY
C        MEMutchler     28-may-83  AUTHOR
C#TYPE    force level io routine
C#FORMAL PARAMETERS   none
C#COMMON BLOCKS
Cio      flage  holds pagebuf
Cin      flhead  holds output text specs
C#METHOD
C  write text to unit keeping track of lines used
C##
```

```
C     FLWTOP*************************************************
$CONTROL check=3,segment=FLRP
      SUBROUTINE FLWTOP
C*                          *** FORMAL PARAMETER DECLARATIONS ***
C*                                                *** ABSTRACT ***
C#PURPOSE writes title line and period header to pagebuf
C#AUDIT HISTORY
C         MEMutchler      28-may-83  AUTHOR
C#TYPE    fore level io routine
C#FORMAL PARAMETERS   none
C#COMMON BLOCKS
Cio       flpage  holds pagebuf
Cin       flhead  holds header text
C#METHOD
C  write text to buf keeping track of lines used
C##
```

```
C     FNDPRD**************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE FNDPRD ( DATE, PERIOD )
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER PERIOD
      INTEGER*4 DATE
      LOGICAL DEARLY
C*                                             *** ABSTRACT ***
C#PURPOSE find number of period to which date belongs
C     like the gpern utility
C#AUDIT HISTORY
C         MEMutchler      31-may-83  AUTHOR
C#FORMAL PARAMETERS
Cin       date    date to look for
Cout      period  number of period to which date belongs
C#COMMON BLOCKS
CIN      flperd   first date each period
C#METHOD  search through datper array untill datper gt date
C  period = iper
C##
```

```
C     GETJOB************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE GETJOB (CLASS, JOBTYP, TOTAL )
C*                      *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE FLPMTR
%INCLUDE FLPERD
      CHARACTER CLASS*10
      CHARACTER JOBTYP*6
      INTEGER TOTAL(MXPERD)
C*                                          *** ABSTRACT ***
C*PURPOSE find all ships in this class having a repair of this
C*       type and adds them all up by period
C         MEMutchler      16 jn 83  AUTHOR
C*TYPE    get info from relate for force level report generator
C*FORMAL PARAMETERS
Cin       class   class name to find repairs for
Cot       total   number of ships repaired in each period
C*OMMON BLOCKS
Cin       flrjob  relate repair relation info
C*CALLER  fljob
C*METHOD
C  get repair job schedule record for latest data date
C**
```

```
C     GETLIF***********************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE GETLIF( CLASS, LIFIND ,ERR,LIFUNT)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER LIFUNT*6,CLASS*10
      LOGICAL ERR
      INTEGER LIFIND
C*                                         *** ABSTRACT ***
C#PURPOSE find standard life of all ships in the class
C#AUDIT HISTORY
C          MEMutchler      31 MY 83  AUTHOR
C#TYPE     get info from relate for force level report generator
C#FORMAL PARAMETERS
Cin        class   class name to find life for
Cot        lifind  standard LIFe IN Days
Cout       lifunt  time units life duration is in
C#OMMON BLOCKS
Cin        shlife  lifetimes for all classes
C#METHOD
C  get standard lifetime for latest data date
C##
```

```
C     GMAKUP**************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE GMAKUP(INPTR,NEXTPTR,FINDX,NUMNEED)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER INPTR,NEXTPTR,FINDX,NUMNEED
C*                                            *** ABSTRACT ***
C#PURPOSE get the record from bgmakup at inptr
C#AUDIT HISTORY
C          MEMutchler      29-JUN-83  AUTHOR
C#TYPE    battlegroup io
C#FORMAL PARAMETERS
Cin       inptr   get this record
Cou       nextptr ptr to next record needed for function makeup
C               =0 if no more there for function
Cou       findx   index into fdefine for this funtion needed by
C                 makeup
Cou       numneed number of this function needed by group makeup
C#COMMON BLOCKS
CIN       gpmkup  holds records to read
C#METHOD
C  Transfer data from the bgmakeup array to the arguments.
C##
```

```
C     PAR2LN*************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE PAR2LN(STRN,LLIN,HAF1,LHAF1,HAF2,LHAF2,ERR)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER LLIN,LHAF1,LHAF2
      LOGICAL ERR
      CHARACTER STRN*(LLIN), HAF1*(LHAF1), HAF2*(LHAF2)
C*                                            *** ABSTRACT ***
C#PURPOSE split line into two parts seperated by a comma
C#AUDIT HISTORY
C         MEMutchler      27-may-83  AUTHOR
C#TYPE    character utility
C#FORMAL PARAMETERS
Cin       line    string to be split
Cin       llin    length of string
Cou       haf1    put first part here
Cou       lhaf1   length of haf1
Cou       haf2    put second part here
Cou       lhaf2   length of haf2
C#COMMON BLOCKS
Cin       charcon  character constants
C#METHOD
C  find a comma and split line by that
C#LOCAL VARIABLES
C         i       index of comma
C##
```

```
C     PAR3LN**************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE PAR3LN(LINE,LLIN,PRT1,LPRT1,PRT2,LPRT2,
     +                  PRT3,LPRT3,ERR)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER LLIN,LPRT1,LPRT2,LPRT3
      LOGICAL ERR
      CHARACTER LINE*(LLIN),PRT1*(LPRT1),PRT2*(LPRT2),PRT3*(LPRT3)
C*                                          *** ABSTRACT ***
C#PURPOS split line into three parts seperated by a comma
C#AUDIT HISTORY
C         MEMutchler      27-may-83  AUTHOR
C#TYPE    character utility
C#FORMAL PARAMETERS
Cin       line    string to be split
Cin       llin    length of string
Cou       prt1    put first part here
Cou       lprt1   length of prt1
Cou       prt2    put second part here
Cou       lprt2   length of prt2
Cou       prt3    put third part here
Cou       lprt3   length of prt3
C#COMMON BLOCKS
Cin       charcon  character constants
C#METHOD
C  find a comma and split line by that
C#LOCAL VARIABLES
C         i       index of coma
C##
```

```
C     READBG*****************************************************
$CONTROL check=3,segment=BGRP
      SUBROUTINE READBG (INFILE, IKEY,LINE,LENLINE,ERR,EOF)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE FLPMTR
        INTEGER INFILE, IKEY, LENLINE
        LOGICAL EOF,ERR
        CHARACTER LINE*LLINE
C*                                               *** ABSTRACT ***
C#PURPOSE reads next non-comment line from file infile, and
C         parses line for firstword<=lkey characters and
C         the rest of the line.  Returns eof=true IFf end of
C         file has been read.  If key=stop is read all lines
C         are ignored untill key=start is read
C#AUDIT HISTORY
C         MEMutchler     27-JUN-83  AUTHO
C#TYPE    read from battlegroup force level report input file
C#FORMAL PARAMETERS
Cin       infile  file from which to read
Cou       key     first word of line read
Cou       lekey  length of key
Cou       line    rest of line read
Cou       lenline length of line read
Cou       eof     end of file flag
C#COMMON BLOCKS
Cin       charcon   global character constants
C#CALER  parsoc
C#METHOD
C  read a line.  If eof then return. If comment line, read again.
C  set key to first non-blank word of line, line to rest and get their
C  lengths
C##
```

```
C     READFL*****************************************************
$CONTROL check=3,segment=FLRP
      SUBROUTINE READFL (INFILE, IKEY,LINE,LENLINE,ERR,EOF)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
%INCLUDE INCPAR
%INCLUDE FLPMTR
        INTEGER INFILE, IKEY, LENLINE
        LOGICAL EOF,ERR
        CHARACTER LINE*LLINE
C*                                             *** ABSTRACT ***
C#PURPOSE reads next non-comment line from file infile, and
C         parses line for firstword<=lkey characters and
C         the rest of the line.  Returns eof=true IFf end of
C         file has been read.  If key=stop is read all lines
C         are ignored untill key=start is read
C#AUDIT HISTORY
C         MEMutchler      16-may-83  AUTHO
C#TYPE    read from input file
C#FORMAL PARAMETERS
Cin       infile  file from which to read
Cou       key     first word of line read
Cou       lekey  length of key
Cou       line    rest of line read
Cou       lenline length of line read
Cou       eof     end of file flag
C#COMMON BLOCKS
Cin       charcon   global character constants
C#CALER  parsoc
C#METHOD
C  read a line.  If eof then return. If comment line, read again.
C  set key to first non-blank word of line, line to rest and get their
C  lengths
C##
```

```
C     SKIPFL***************************************************
$CONTROL check=3,segment=FLBG
      SUBROUTINE SKIPFL(INFILE,ERR,EOF)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
        INTEGER INFILE
        LOGICAL EOF,ERR
C*                                            *** ABSTRACT ***
C#PURPOSE reads and ignores all lines utill a start line
C         is read
C#AUDIT HISTORY
C         MEMutchler      16-may-83  AUTHOR
C#TYPE    read from input file
C#FORMAL PARAMETERS
Cou       err     an error was found
Cou       eof     end of file flag
C#COMMON BLOCKS
Cin       charcon   global character constants
C#CALLER  parsoc
C#METHOD
C  read a line.  If eof then return. If comment line, read again.
C  set key to first non-blank word of line, line to rest and get their
C  lengths
C##
```

12.0 MANUAL ASSIGNER MODULE

12.1 PURPOSE

The assigner provides the user with a high-level facility for creating and modifying ship construction schedules. A typical five-year shipbuilding program projection contains more than 100 ship schedules; a typical schedule record from one of the ncjodat relations contains perhaps 20 data fields. This represents a fairly large volume of data.

One of the principal activities of program analysis is program redesign, which involves changing the number of ships to be built, their timing, and/or which shipyards will perform the jobs. Creating a new program or making changes is very time consuming if done on a schedule-by-schedule basis given the amount of data the raw schedules contain. Also, analysts often prefer to perform this activity by outlining the broad pattern of the program, e.g. the number of ships of each class to be awarded each year, rather than by working with the detail of the schedules.

The assigner is a productivity tool designed to support this usage pattern. It is a specialized editor which presents the user with shipyard assignments by yard, ship class, job type, and period in a tabular fashion, and which accepts changes to the assignments. At the close of a session it will create a new set of schedule records as implied by the assignments (using construction job descriptions for each class in its computation of schedule record fields), and will write them into the ncjodat.projj relation as the current schedules for the user's scenario. Shipbuilding program schedule creation and modification is thus a quick and intuitively natural process, rather than a tedious one in which the analyst can become lost in detail.

A number of features flesh out this basic capability. The user may choose to edit only a subset of the schedules, with the subset being defined by the yard, class, and job type code names that are "on" in the assigner's Command System list menus. Individual schedules changed using the DBU may be marked as unchangeable by the assigner so that the field values the user specified are not arbitrarily overwritten. The assigner checks for the existence of appropriate job description data whenever assignments are added or modified; if they are not found, the user may put the assigner on hold and return to the Command System and the DBU to enter the job description, and then come back to finish his assigner session. Command system parameters give the user the capability to configure the assigner in various ways. For example, the algorithm which computes new schedules from the rather sketchy data on the display page may be "tuned" by setting parameter values.

12.2 SUMMARY OF STRUCTURE

The assigner is structured in three major parts, as shown in Figure 12-1. This corresponds to the three fairly separable tasks which it must perform. When invoked, the assigner must first read the schedules in the data base for the current scenario and convert these into a form usable during the editing phase. Then it must support user editing by offering a variety of interactive command options. When the user is finished, it must compute new schedules and save them in the data base.

This division into parts is implicit, showing up only in the flow of execution of the asgn.prog FORTRAN program which implements all three. This program is run by the Core as a son process; its handling somewhat resembles the DBU in that the user may return to the Command System from this process without terminating it, coming back to reativate the process and resume his in-progress editing session later.

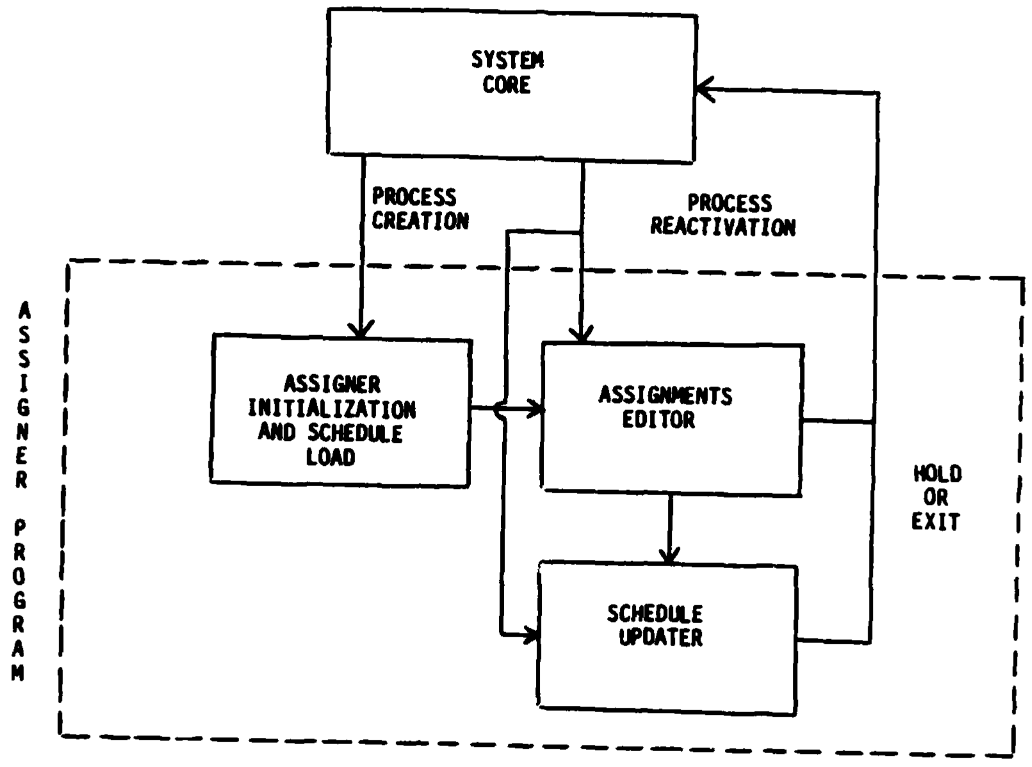


Figure 12-1. Assigner Flow of Control Structure

Because of the variety of its functions the assigner has a particularly rich data structure. The center of this data structure is two direct-access binary files which the assigner creates in the user's log-on group (called bufasn and cmnasn) in which all assignment records (an assignment record corresponds to a single row appearing on the display screen) and the values of important common blocks are saved. This buffering of the data both conserves data memory (crucial on the HP) and provides abort protection. If an abort occurs for any reason during an editing session the user can always recover to that point, since the assigner automatically looks for and offers the user the option of using an existing bufasn/cmnasn during its initialization. The user need only re-run the assigner to effect recovery.

Bufasn and cmnasn and the /asgn/ and /asnvld/ common blocks are the primary means of communication between the three parts of the program. The next sections will discuss the structure of each part in more detail.

12.2.1 Terminology

Before continuing it is necessary to define some terms and concepts which will be used throughout this section. They are:

1) ASSIGNMENT: An assignment is a count of 1 appearing in any row and column of the assigner display. It is the fact that a given ship will have a given job done at a given yard in a given period. A bufasn record element or a display page cell (a row/column location) of "3", for example, denotes 3 assignments.

2) ASSIGNMENT RECORD: A row of assignments, or a bufasn record.

3) SCHEDULE: A record with specific milestone dates for a given ship job, in the form used in the ncjodat and rejodat relations. A schedule is a detailed version of an assignment.

4) TUPLE: A schedule record that is resident in one of the ncjodat or rejodat relations.

5) CLASS-JOB: Any assignment will be to perform a given job on a ship of a given class in a given yard. An assignment record is all assignments for that job on that class in that yard. On the display page, assignment rows are labeled by the class name and a single-character code indicating the job type (blank indicated new construction). A class-job is such a row or the label on the row.

6) JOB SERIES TYPE: Assignments for a given class-job can be characterized as lead-ship jobs, first-in-yard jobs, follow ship jobs, etc. What variety, or job series type, a given assignment belongs to is represented on the display page by a single-character code appearing at the location of the assignment display cell. An "L2" indicates 2 assignments in the given period, the first of which is a lead ship.

7) INBOUND/OUTBOUND: The initialization phase of assigner execution is sometimes referred to as the "inbound leg", while the DB update phase is sometimes called the "outbound leg". The inuition of the terms is based on the direction of flow of data between the assigner and the data base.

8) HARD-WIRED TUPLES: The user may specify that a given schedule tuple not be changed during the assigner's DB update step by setting the AUTOMOD field of the tuple to a "NO" value in the DBU. This is a no-assigner-modify or hard-wired tuple, one which the DB update logic must not change in any way (except that it can be deleted if the user deletes all the assignments for its class job in its period).

12.2.2 Initialization Structure

Figure 12-2 diagrams the structure of the initialization phase, with an emphasis on data flow. Initialization is triggered on asgn process creation and first activation. It is not repeated if the user puts the assigner on hold during the editing phase and then returns to it later; the user is just back where he was when he left in that case.

Figure 12-3 summarizes the flow-of-control of the initialization step. Although several data structures contribute to intialization, the central goal and activity of the process is the read of data base schedules and conversion of them into assignment records.

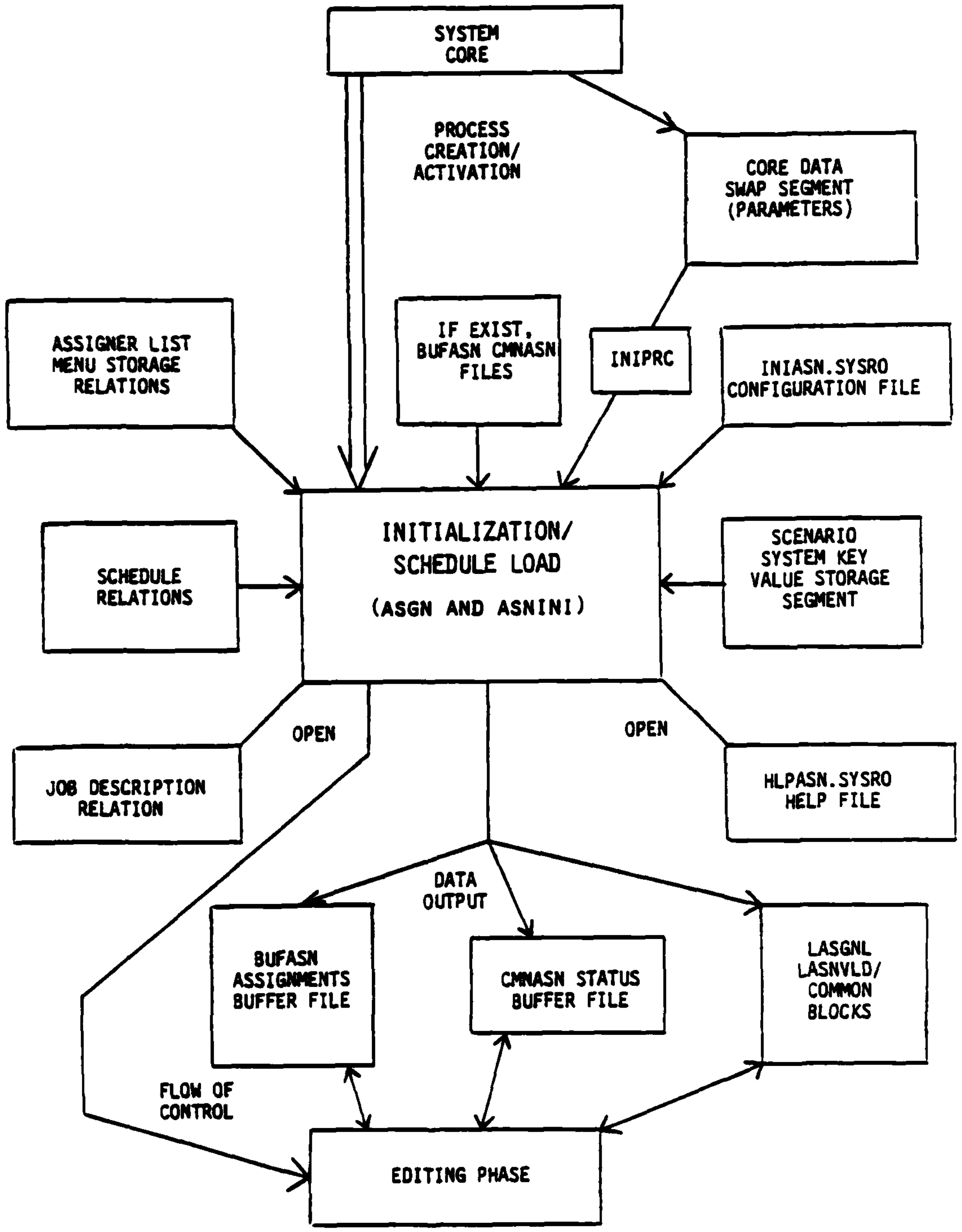

SYSTEM CORE
PROCESS CREATION/ ACTIVATION
CORE DATA SWAP SEGMENT (PARAMETERS)
ASSIGNER LIST MENU STORAGE RELATIONS
IF EXIST, BUFASN CMNASN FILES
INIPRC
INIASN.SYSRO CONFIGURATION FILE
INITIALIZATION/ SCHEDULE LOAD (ASGN AND ASNINI)
SCHEDULE RELATIONS
SCENARIO SYSTEM KEY VALUE STORAGE SEGMENT
OPEN
JOB DESCRIPTION RELATION
OPEN
HLPASN.SYSRO HELP FILE
DATA OUTPUT
BUFASN ASSIGNMENTS BUFFER FILE
CMNASN STATUS BUFFER FILE
LASGNL LASNVLD/ COMMON BLOCKS
FLOW OF CONTROL
EDITING PHASE

Figure 12-2. Assigner Initialization Structure

Figure 12-3. Narrative Summary of Assigner Initialization

1. Call iniprc to swap in Core data.

2. Check to make sure user has write priveleges to ncjodat.projj

3. Set flags.

4. Open ncjdat.descj to support dynamic checking of job description availability as user enters new assignments.

5. Open and read iniasn.sysro

6. Open bufasn if it exists in the log-on group, or create it.

7. Open the help text file hlpasn.sysro

8. Open cmnasn if it exists in the log-on group, or create it.

9. If bufasn and cmnasn existed, see if the user wants to use them or start fresh. If use, initialize system to its old state from cmnasn's contents, bring in the lists of valid yards/classes/jobtypes for this invocation, and flush any invalid assignments records from the old bufasn. Write the display screen and we're done.

10. If bufasn/cmnasn didn't exist or the user want to start fresh, bring in the lists of valid yards/classes/jobtypes, and read the data base for schedules, converting them into assignments records. Write the display screen.

On Figure 12-2 note in particular the read of the iniasn.sysro configuration file. The assigner is very much a data-driven system, with many important elements appearing as variables rather than being hard-wired into the code. Many of these variables are set by reading iniasn, making it easy to change them as appropriate.

12.2.3 Editor Structure

As befits an editor, the second part of the assigner is fundamentally organized around obtaining and responding to user commands. The commands are summarized in Figure 12-4.

They can be divided into three types: paging commands which let the user look at a different time frame or set of shipyards/classes, assignment-modification commands which implement the basic editing functions of add, delete, modify, and copy, and service commands such as help requests and exit requests.

Figure 12-5 presents a typical assigner display page. The page is effectively a window on the assignments records held in the bufasn file. The window is up to 15 lines long and twenty columns (periods) wide. The position of the window is changed by the page up/down (+, -, ++, --) and the page right/left (>, <, >>, <<) commands. Paging never changes the assignments; the paging algorithm merely causes a different part of the buffer to be extracted and printed to the screen.

The modification commands (A, I, D, M, R, and RC and their permutations) do alter the assignments records by addition, deletion, modification, or copying. Most require the user to specify by number both a yard and a class-job to be changed. In figure 12-5, the LSD-49 assignments at Avondale would be indicated by the number 1.2. Most prompt the user for the new or changed assignments, perform basic data validation on the

Figure 12-4. Summary of Assigner Editing Commands

Command	Description	Command	Description
?	= Obtain help from a menu	? #	= Print help subject number #
	= Refresh assign display	^	= EXIT assignments module
[	= Display previous page	[[	= Display topmost page
]	= Display next page	]]	= Display last page
<	= Display left neighbor	<<	= Display leftmost page
>	= Display right neighbor	>>	= Display rightmost page
A	= Add a new yard	A #	= Add new shipclass to yard #
I #	= Add new yard before #	I #.	= Add new class before #.
D #	= Delete an entire yard	D #.	= Delete class from yard #
MN #	= Modify Name of yard #	MN #.	= Modify Name of class #.
P	= Display to line printer	P #,	= Print from yard # to on LP
M #.	= Modify assignments for class in yard #		
R #	= Relocate yard numbered # to end of list		
R #,##	= Relocate yard # to before yard number ##		
R #.,##	= Relocate #. to end of yard ##'s classes		
R #.,##.	= Relocate class #. to before class ##.		
RC # / RC #,##	= Like R, except copy yard instead of move		
RC #.,## / RC #.,##.	= Again like R; copy class instead of move		

Figure 12-5. A Typical Assigner Display Page

```
Scenario: DEMO               *SHIP ASSIGNMENTS*   Page  1A   Time in: FISCYR
Yard    Period:| 1  2  3  4  5  6  7  8  9                                 |  9
    Shipclass T|86 87 88 89 90 91 92 93 94                                |TOT
AVONDALE    #01|--+--+--+--+--+--+--+--+--|--------------------------------|---
  1 LSD-41     | 2  1                                                      |  3
  2 LSD-49     |      L2  2  2                                             |  6
  3 T-AO-187   | 2  2  2  2  2                                             | 10
BIW         #02|--+--+--+--+--+--+--+--+--|--------------------------------|---
  1 CG-47      | 1  1  1  1                                                |  4
  2 DDG-51     |   Y1 F2  1  2                                             |  6
EB GROT     #03|--+--+--+--+--+--+--+--+--|--------------------------------|---
  1 SSBN-726   | 1  1  1  1  1                                             |  5
  2 SSN-21     |         L1                                                |  1
  3 SSN-688    | 2  2  2  1  2                                             |  9
GDQ         #04|--+--+--+--+--+--+--+--+--|--------------------------------|---
  1 AE         |      Y1  1  1                                             |  3
  2 AG         |L1                                                         |  1
  3 AO-187    c|       1  1                                                |  2
               |--+--+--+--+--+--+--+--+--|--------------------------------|---
 14  33  TOTALS|29 24 33 30 27     1                                       |144
(?=help) >
```

response, and alter the contents of bufasn and/or the /asgn/ common block in response.

The service commands provide miscellaneous functions such as help, module exit, and sending of the assigner display pages to a printer.

At its highest level the structure of the editor is extremely simple, as indicated in Figure 12-6. The asgn program unit routine calls a routine which prompts the user for a command and which decode the response, and then calls the executive routine for the given command. Complexities in the editor implementation involve the details of executing particular commands; the complexities are "pushed down" into subsidiary routines (which will be discussed in Section 12.5).

12.2.4 Data Base Update Structure.

The task of the data base update logic is conversion of a summary description of program schedules, the assignments, into detailed schedules in the ncjodat.projj relation. This involves generation of more detailed than is explicitly contained on the assigner display page. The detail is reconstructed using information from the new construction job schedule descriptions, or planning factors, found in the ncjdat.descj relation, and by applying rules of thumb.

The principal computational task is generation of the schedule dates. A single date can be inferred for an assignment from the column on the assigner display in which it appears; the rest must be calculated from this "basis date" using the milestone-to-milestone time intervals given in ncjdat.descj. The user may specify use of various date-spreading algorithms (e.g., compute the schedules such that all starts in a given yard for a given class are evenly spaced over time) by setting parameters in the Command System menu.

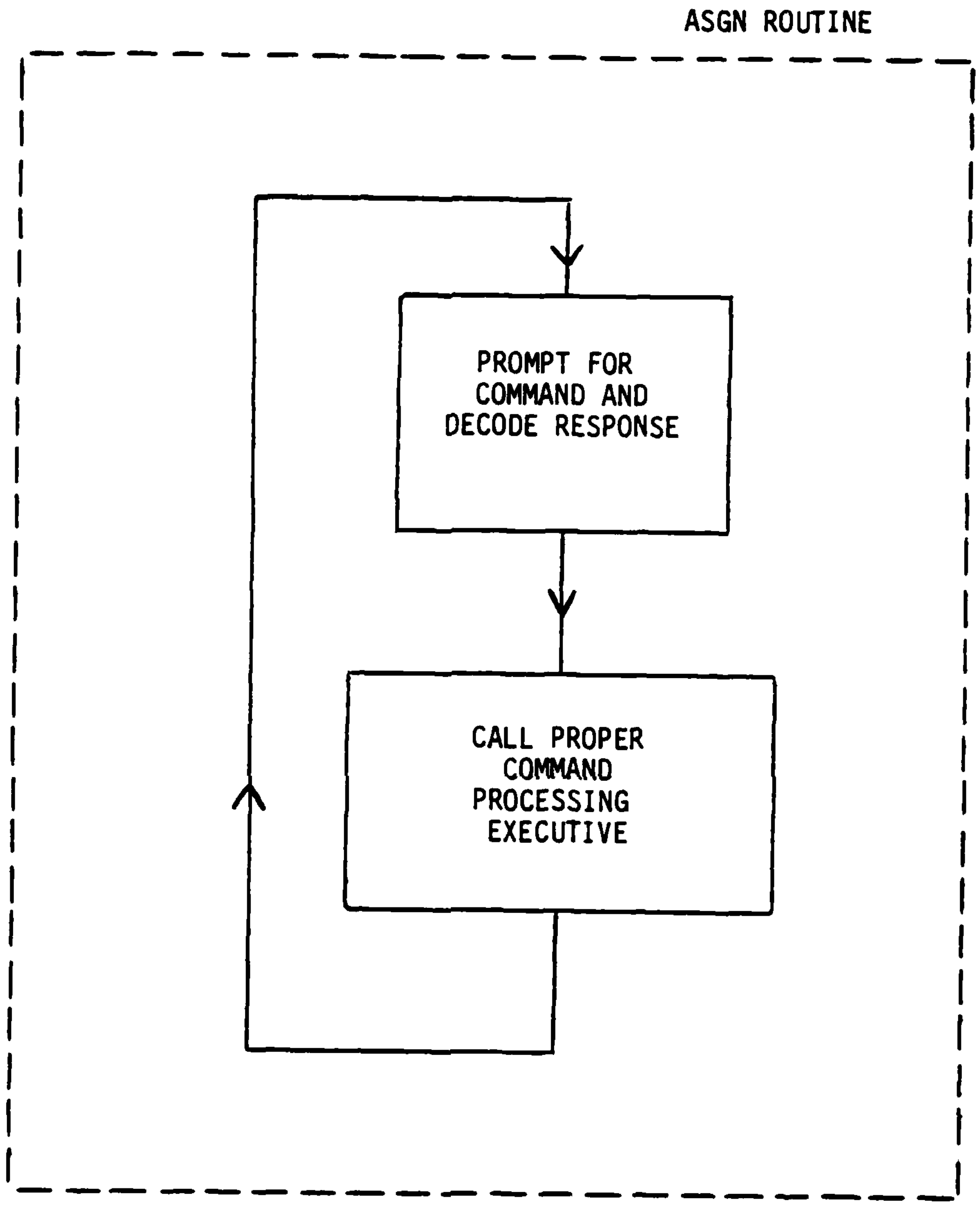

ASGN ROUTINE
PROMPT FOR COMMAND AND DECODE RESPONSE
CALL PROPER COMMAND PROCESSING EXECUTIVE

Figure 12-6. Assigner Editor Structure

The overall structure of the update step is shown in Figure 12-7. The user may have assignments for ships whose schedules are in the historical and current schedule relations displayed by setting parameters in the Command System menu; these must be removed before new projected schedules are generated or too many will be created.

It is fairly common for users to enter assignments for which no job description is available in ncjdat.descj. When the schedule creation logic detects this problem it tells the user about it and returns him to the Command System, putting the assigner process on "Hold" so he can enter the required data using the DBU and come back to the assigner to finish the data base update.

Schedule records must be given hull numbers only after the write of the new schedules to ncjodat.projj is complete because of the option which lets the user mark schedules as unchangeable by the assigner (AUTOMOD="NO"). Ncjodat.projj records must have unique values for the key SCENARIO,CLASS,HULL,COMNUM; aborts can occur if the update logic assigns final hull numbers before the write since a no-assigner-mod record might have the same hull number as one the assigner attempts to add (RELATE unary key index violation results). Records are written with negative hull numbers, and these are then changed by a logic which takes the presence of no-assigner-mod records into account.

The structure for the actual schedule generation and update step is pictured in more detail in Figure 12-8. This complex task is organized around the requirements of the date-spreading algorithm, which requires as input an ordered list of the candidate ship schedule dates (one per ship) as generated from assignments' column position on the display screen. The user may specify no spreading, spreading within a class-job, or spreading within a compexity-group. For example, even intervals between starts of DDG-51 construction jobs at BATH might be desired

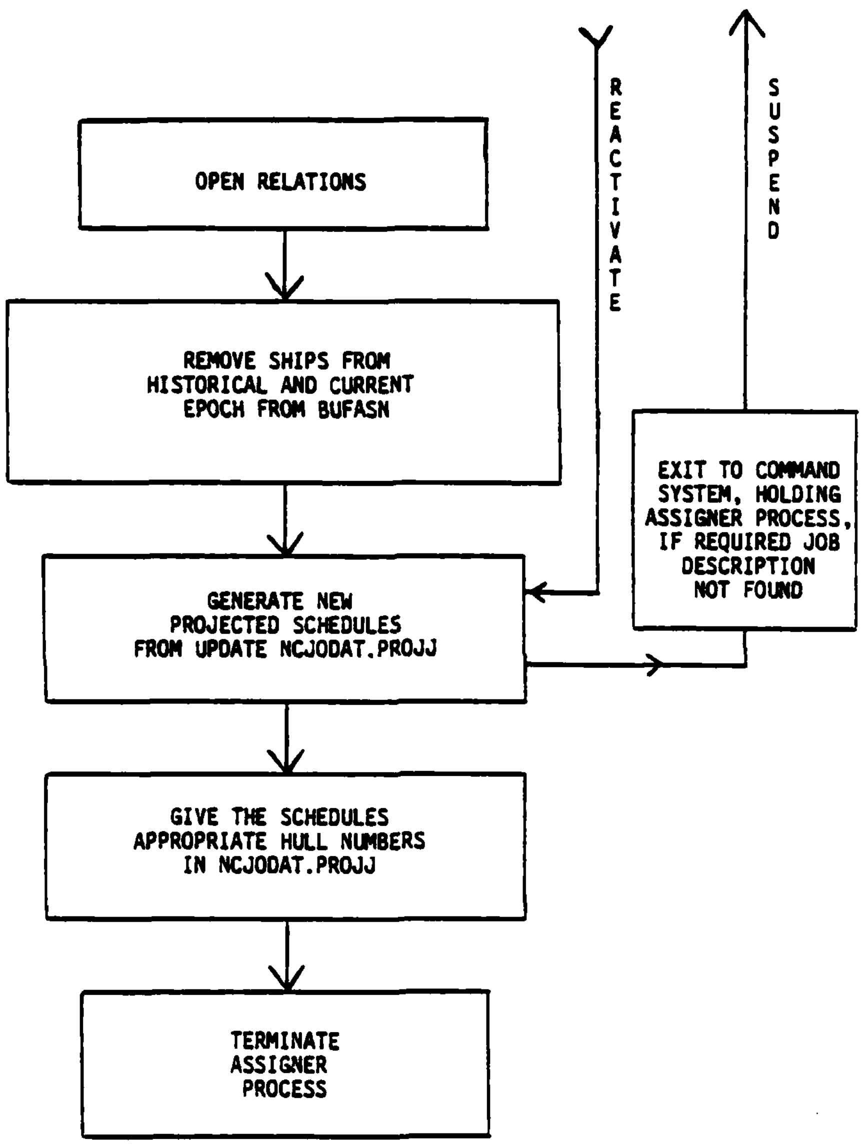

REACTIVATE
SUSPEND
OPEN RELATIONS
REMOVE SHIPS FROM HISTORICAL AND CURRENT EPOCH FROM BUFASN
EXIT TO COMMAND SYSTEM, HOLDING ASSIGNER PROCESS, IF REQUIRED JOB DESCRIPTION NOT FOUND
GENERATE NEW PROJECTED SCHEDULES FROM UPDATE NCJODAT.PROJJ
GIVE THE SCHEDULES APPROPRIATE HULL NUMBERS IN NCJODAT.PROJJ
TERMINATE ASSIGNER PROCESS

Figure 12-7. Assigner Data Base Update Overall Structure

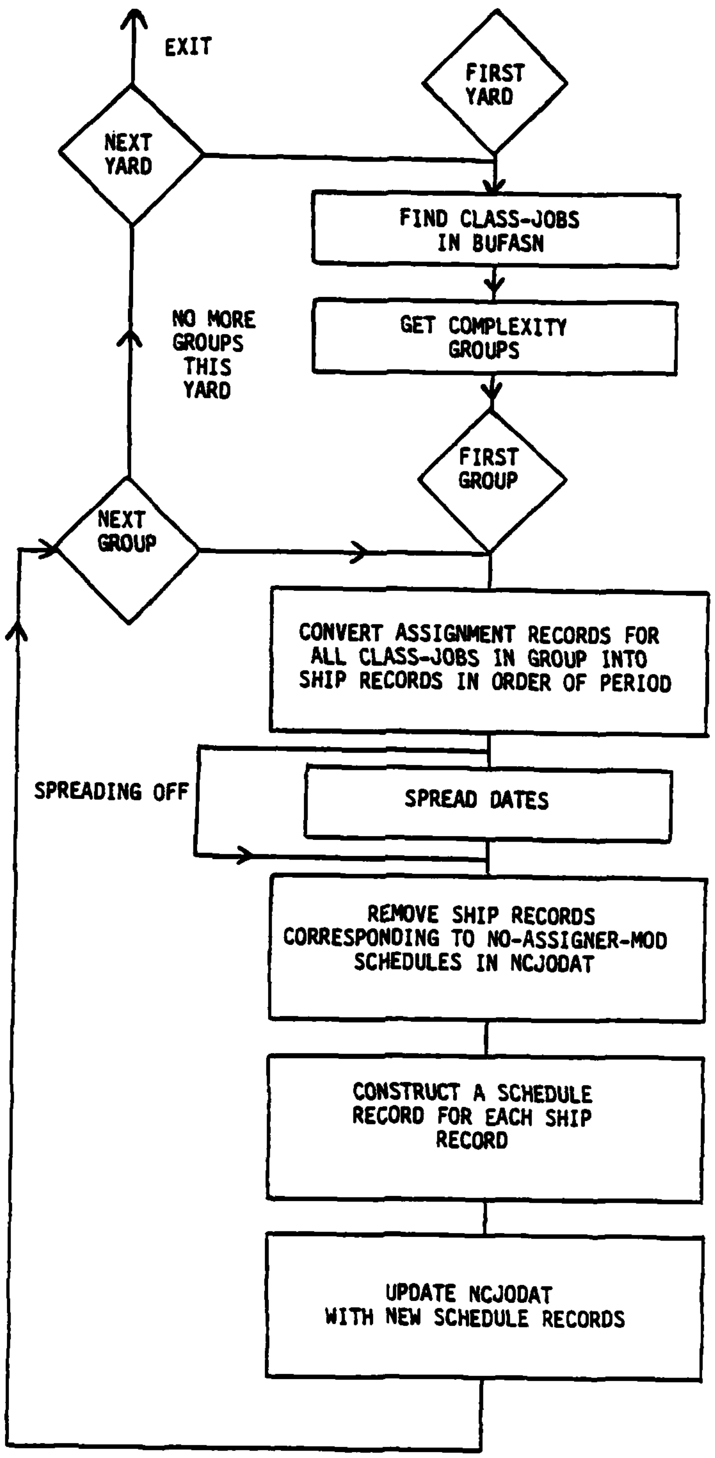

EXIT
FIRST YARD
NEXT YARD
FIND CLASS-JOBS IN BUFASN
NO MORE GROUPS THIS YARD
GET COMPLEXITY GROUPS
FIRST GROUP
NEXT GROUP
CONVERT ASSIGNMENT RECORDS FOR ALL CLASS-JOBS IN GROUP INTO SHIP RECORDS IN ORDER OF PERIOD
SPREADING OFF
SPREAD DATES
REMOVE SHIP RECORDS CORRESPONDING TO NO-ASSIGNER-MOD SCHEDULES IN NCJODAT
CONSTRUCT A SCHEDULE RECORD FOR EACH SHIP RECORD
UPDATE NCJODAT WITH NEW SCHEDULE RECORDS

Figure 12-8. Schedule Creation and Update Structure

(within class-job), or perhaps even intervals between starts for the combination of DDG-51 and CG-47 jobs (complexity-group option). A complexity group is defined as any set of class-jobs in the same yard with the same value in the COMPLXGRP field in their ncjdat.descj job description records.

Once the dates are spread, producing final schedule basis dates for each ship, the "hard-wired tuple removal process" look through ncjodat.projj for tuples in the current scenario with AUTOMOD field values of "NO", finds the corresponding ship-record for each one found, and removes that ship record so no double counting occurs. The match-up between tuples and ship-records is done according to schedule basis date.

Then complete schedule records are constructed for each ship-record and are placed in a temporary direct-access file used as a buffer. The ncjodat.projj update logic then takes records from this file and either updates them over corresponding existing tuples in ncjodat, adds them to ncjodat, or deletes ncjodat tuples when there are more of those than there are schedule records.

12.3 INPUTS AND OUTPUTS

The assigner's principal outputs are the display screens presented to the user during editing (such as the sample presented in Figure 12-5), and updated schedules in the ncjodat. projj relation. The primary purpose of the screens is to support interactive editing, but they also can serve as "final" outputs when the user causes them to be printed by giving the "P" command.

The schedules which are output have the following characteristics:

1) The milestone dates they contain are consistent with both the time pattern of assignments which was showing on the display screens when the 'Q'/'^' command was given, and with the schedule planning factors read from the job description relation ncjdat.descj. When the display time units are years and the date spreading basis date iS not appropriation or award, the award date will be on the month and day specified in the DEFLTAWDAY field in ncjdat.descj. When DEFLTAWDAY is used, the schedule typically cannot agree completely with both it and the milestone-to-milestone intervals. The emphasis in these cases is on computing appropriation/award dates such that subsequent runs of the assigner will show the same time pattern of assignments.

2) The YARD, CLASS, NCJOBT, and JSTYP fields all have values consonant with the names and code characters which were showing on the display. COMNUM is always set to 1.

3) DATADATE and ENTRY_DATE are both set to the current date, ENTRY_BY to the current user, and DATASOURCE to 'ASSIGNER'.

4) HULL is set by the complex algorithm implemented in the newhul.rprocs RELATE execute file. The algorithm tries to create hull numbers which continue the sequences found in the historical or current schedule relations.

5) All other fields are set according to the specifications of the job description records.

An example of some of the schedules created by the assigner from the AVONDALE assignments shown in Figure 12-5 is shown in Figure 12-9.

Figure 12-9. Sample Schedules From Ncjodat.projj Relation

```
$LINE SCENARIO     CLASS      HULL COMNUM YARD     NCJOBT JSTYP  CUSTOMER
SHIPNAME                       CMETHD     APPROP
     AWARD      START       KEEL     LAUNCH   DELIVERY COMMISSION DAYSADDED
ASNORDER   DATADATE DATASOURCE ENTRY_BY ENTRY_DATE AUTO
PROGVAR1 PROGVAR2 SUBRELUMAP

  334 DEMO         LSD-41        42       1 AVONDALE NEWCON ORDFOL USN
MODULZ 10/01/1985
11/01/1985 11/01/1986  5/01/1987  1/01/1989  5/01/1990  6/01/1990         0
57553221 10/28/1984 908/SHAPM  MARK     10/28/1984 YES
      42        0          0
  335 DEMO         LSD-41        43       1 AVONDALE NEWCON ORDFOL USN
MODULZ 10/01/1984
11/30/1984  6/30/1986                         1/30/1989                   0
0  9/01/1984 908/SHAPM  DBA       9/25/1984 YES
      41        1          0
  336 DEMO         LSD-41        44       1 AVONDALE NEWCON ORDFOL USN
MODULZ 10/01/1985
11/01/1985  7/02/1987  1/02/1988  9/02/1989  1/02/1991  2/02/1991         0
57553221 10/28/1984 908/SHAPM  MARK     10/28/1984 YES
      43        0          0
  337 DEMO         LSD-41        45       1 AVONDALE NEWCON ORDFOL USN
MODULZ 10/01/1986
11/01/1986  3/01/1988  9/01/1988  5/01/1990  9/01/1991 10/01/1991         0
57553221 10/28/1984 908/SHAPM  MARK     10/28/1984 YES
      44        0          0
  338 DEMO         LSD-49        49       1 AVONDALE NEWCON LEAD   USN
MODULZ 10/01/1987
11/01/1987 11/01/1988  7/01/1989  6/01/1991  6/01/1993  7/01/1993         0
57553222 10/28/1984 908/SHAPM  MARK     10/28/1984 YES
      49        1          0
  339 DEMO         LSD-49        50       1 AVONDALE NEWCON ORDFOL USN
MODULZ 10/01/1987
11/01/1987  5/02/1989  1/02/1990 12/02/1991 12/02/1993  1/02/1994         0
57553222 10/28/1984 908/SHAPM  MARK     10/28/1984 YES
      50        0          0
  340 DEMO         LSD-49        51       1 AVONDALE NEWCON ORDFOL USN
MODULZ 10/01/1988
11/01/1988 11/01/1989  7/01/1990  6/01/1992  6/01/1994  7/01/1994         0
57553222 10/28/1984 908/SHAPM  MARK     10/28/1984 YES
      51        0          0
  341 DEMO         LSD-49        52       1 AVONDALE NEWCON ORDFOL USN
MODULZ 10/01/1988
11/01/1988  5/03/1990  1/03/1991 12/03/1992 12/03/1994  1/03/1995         0
57553222 10/28/1984 908/SHAPM  MARK     10/28/1984 YES
      52        0          0
```

Inputs the assigner requires include the iniasn.sysro configuration file, the Core data swap segment (with its current values from the assigner's parameter menu), the lists of yards, classes, and job types of interest for the run from the assigner's Command System list menus, the schedules found in the data base for the given scenario at the time of execution, the job description relation's contents for the given scenario, scenario key field values from the scenario system's extra data segment, and (of course) editing-session commands and inputs from the user.

12.3.1 The Configuration File and the Help File

Figure 12-10 shows a copy of the assigner configuration file stored in iniasn.sysro. This file is read during initialization to discover the values of certain assigner operating parameters. The file contains six lines, read sequentially using FORTRAN formatted i/o. The formats are typically fixed, so that column position of the data is important. The contents of the lines are as follows:

LINE 1: Six integers. The first may be 0 or 1, specifying whether the main intialization and editing lprnt (number 7) will be off or on. The second is the FORTRAN unit number for diagnostic output, typically the same as standard output. If changed, be sure to give the proper FILE equations prior to assigner execution so that diagnostics go to the proper device. The third number may be 0 or 1, specifying whether subsequent reads from iniasn will not or will be echoed. The fourth number specifies the FORTRAN unit number the echo will be sent to (same warnings apply as for the diagnostic unit). The fifth and sixth numbers specify the unit numbers for normal interactive input and output. The text string to the right of the 6 I5 fields is a brief reminder of the purpose of each field.

LINE 2: An integer, a *2 character string, and an up to *16 character string. The *2 string is the short interactive command prompt, set to "> " in the example. The large string is the long interactive prompt, to which the short prompt is concatenated.

Figure 12-10. Text of INIASN.SYSRO Configuration File

```
     0    6    0    6    5    6      (6I5) iprnt,ioutp,iecho,uecho,in,out
      9½     (?=help)                "½ " short, "(?=help) " long
    10       LF               ncrs        asncas,cdchar,jtchar-"n"=default
   BUFASN                     1  500
   HLPASN.SYSRO               2
   CMNASN                     3
```

The number, given before the strings, is the number of characters in the long string. The command prompt is thus very easily configurable.

LINE 3: A single number specifying user input uppercasing rules for yard and class names. The remainder of the fields on the line are obsolete and ignored. The number specifies a bit map where bits 1 and 3 on indicate uppercasing only of the first characters of names, bits 2 and 4 uppercasing of all characters in names (the job type code character at the end of a class-job name is lowercased later in the current logic). The setting of 10 invokes complete uppercasing.

LINE 4: The name, FORTRAN i/o unit number, and maximum number of records allowed in the bufasn assignments holding buffer file. The file will be created in the log-on group and i/o will take place through the given unit number. Should 500 class-jobs ever be insufficient the limit can easily be expanded by changing the third parameter of this line.

LINE 5: The name and FORTRAN i/o unit number of the prthlp-readable assigner help text file, currently hlpasn.sysro.

LINE 6: The name and FORTRAN i/o unit number for the cmnasn file which is companion to bufasn. It always has only two records, since it contains the contents of two of the common blocks in the /asgn/ include file. It's primary function is to support recovery after an abort by holding the system status at the time of the abort.

The great majority of on-line help for the assigner is stored in the hlpasn.sysro file. This is a standard EDITOR-type ASCII file divided into sections by the %BEGIN statements recognized by the prthlp utility. This file is opened during initialization and read as necessary in response to user help requests.

12.3.2 Variables From the Assigner's Command System Parameter Menu

The assigner actually uses a good deal of the data in the swap segment provided by the Core and read during initialization by a call to the iniprc utility. Of interest are lprnts settings and the contents of the /scenar/ and /uzrprv/ common blocks. Of principal interest are the setting of variables on the assigner's parameter menu. These are read by asnlbs, which transfers the values from their storage locations in /pvalue/ to variables in /asgn/. A sample of the assigner parameter menu is shown in Figure 12-11. The meaning and use of each parameter is:

1) TIME UNIT: Specifies the amount of time that each display page column represents. When combined with STARTING DATE, also determines the first day of the period each column represents.

2) STARTING DATE: First day of the period the user wants assignments displayed for. It is permissible to work with a subset or superset of the periods represented by schedules currently in ncjodat.projj. If the date given is not the first day of the period type specified it is moved back to that day (i.e. if fiscal years is the TIME UNIT and the STARTING DATE is given as 1/1/1986, it will be moved back to the first day of fiscal 1986, 10/1/1985).

3) ENDING DATE: Last day of the time span of interest. Determines the number of columns on the display page in combination with the first two parameters.

4) CANDIDATE SHIP YARDS: A gate to a list menu with the names of all the shipyards ALIAS knows about. The user can work with assignments for only a subset of these yards by setting some of the list names to 'OFF' status. No assignments are loaded for the 'off' yards, and their schedule records in ncjodat.projj (if any) are not updated.

5) CANDIDATE SHIP CLASSES: Like candidate yards, lets the user work with a subset of ship classes only.

6) CANDIDATE JOB TYPES: Like the previous two, lets the user see and update only class-jobs of certain job types.

7) DISPLAY BASIS: The column any given assignment is placed in depends on which schedule milestone date is being used as the basis for making assignments. For any given schedule, a different column will typically be chosen if the basis is DELIVERY rather than AWARD. The setting of this parameter determines which milestone is used as the basis date.

8) ADJUST BASIS: The schedule milestone date being used as the basis for date-spreading. Though the user may specify assignments in terms of AWARDs (display basis), he may want, e.g., the start dates of the resulting schedule evenly spaced over time.

Figure 12-11. Sample Assigner Parameter Menu

```
Menu is ASNPRM             * ALIAS COMMAND SYSTEM *   Scenario is DEMO
-------------------------------------------------------------------------------
               MANUAL ASSIGNER MODULE INITIALIZATION PARAMETERS
 1.  TIME UNIT                 = FISCYR       (FISCYR,CALYR,QTR,MONTH,WEEK,DAY)
 2.  STARTING DATE             =  1/ 1/1980   (MM/DD/YYYY)
 3.  ENDING DATE               = 12/31/1999   (MM/DD/YYYY)
 4.  CANDIDATE SHIP YARDS      = LIST         (ALL/LIST)
 5.  CANDIDATE SHIP CLASSES    = LIST         (ALL/LIST)
 6.  CANDIDATE JOB TYPES       = LIST         (ALL/LIST)
 7.  DISPLAY BASIS             = AWARD        (APPROP,AWD,START,KEEL,LNCH,DELIV)
 8.  ADJUST BASIS              = START        (APPROP,AWD,START,KEEL,LNCH,DELIV)
 9.  ADJUST MODE               = PROGRAM      (NONE,PROGRAM,COMPLX-GROUP)
10.  JOBS EPOCH OPTION         = PROJ         (ALL,CURR/PROJ,PROJ)
11.  SHIPCLASS SORT ORDER      = ALPHABETIC   (ALPHABETIC,INPUT ORDER)
12.  SHIPYARD SORT ORDER       = INPUT ORDER  (ALPHABETIC,INPUT ORDER)
13.  AUTO REFRESH              = OFF          (ON,OFF)

  COMMAND:
```

9) ADJUST MODE: This setting controls the operation of the date-spreading algorithm employed during schedule generation. If NONE, then aspred is simply never called. If PROGRAM, aspred is fed only the ships for a single class-job in a single yard when it is called. If COMPLX-GROUP, aspred is fed all class-jobs in a yard in the same complexity group, where complexity group is specified by the COMPLXGRP field value in the ncjdat.descj relation. The latter case might be desirable when a yard is building similar ships of different classes, e.g. DDG's and CG's.

10) JOBS EPOCH OPTION: Controls which relations are read for schedules during initialization. PROJ is the normal setting since only projected schedules can be updated anyway. Note that if the setting is not PROJ then asclen must be called during the outbound leg, at a substantial processing penalty.

11) SHIPCLASS SORT ORDER: The user may specify that class-jobs be listed alphabetically within a given yard on the display screen, or in the order in which they were input or displayed during the last session.

12) SHIPYARD SORT ORDER: Similar to the previous parameter, but its setting has no effect at this time. The display order is always alphabetic.

13) AUTO REFRESH: If ON, the assigner display will be refreshed (rewritten) every time the user gives a command which changes its contents or writes substantial output to the page. If OFF, the user must always request a refresh via the command.

12.3.3 List Menus

The assigner's list menus were alluded to in the last section. Figure 12-12 shows the third of them, the CANDIDATE JOB TYPES menu. The user can restrict the types of job for which assignments will be read from the data base and displayed; this restriction also prevents the user from entering any new assignments of the "off" job types. Note that new job type codes added to the system must be explicitly added to a user's scenario with the NC_JOB_TYPES and RE_JOB_TYPES screens of the DBU before they will appear on this list menu, and thus before their assignments can be displayed.

Figure 12-12. Sample Assigner Valid Job Types List Menu

```
Menu is CHJTYP            * ALIAS COMMAND SYSTEM *   Scenario is DEMO
--------------------------------------------------------------------------------
              CHOOSE THE SET OF VALID JOBS WHICH MAY BE ASSIGNED
--------------------------------------------------------------------------------
    1. * CONV                            5. * REPAIR
    2. * NEWCON                          6. * SLEP
    3. * REACT                           7. * SLPCNV
    4. * REFUEL
--------------------------------------------------------------------------------

  COMMAND:
```

The lists are stored in the valcls, valyds, and vljtyp relations in the .mnurel group (.makmenu for the development system).

12.3.4 Sample Schedules

A sample of ncjodat.projj schedule records was shown in Figure 12-9. For input purposes the assigner is only interested in a restricted set of the fields: SCENARIO, YARD, CLASS, HULL, COMNUM, NCJOBT, JSTYP, DATADATE, ENTRY_DATE, and the particular milestone date field being used as display basis (e.g. AWARD) are the only ones read. DATADATE and ENTRY_DATE are consulted during reads of the historical and current relations to ensure that no double-counting occurs due to multiple reads of the same schedule for different data dates.

12.3.5 Job Description Records

Figure 12-13 shows sample job schedule description records from ncjdat.descj, which are read during the DB update phase in order to gather information necessary to construct complete schedule records from the assignments. Note particularly that the YARD field may take on the name of a specific yard or ANY; the assigner always searches for a match on the name of the yard an assignment is in first (along with matches on the other keys, of course), but will take any ANY record if the first search fails.

12.3.6 Scenario Key Field Values

The assigner makes use of the scenario system via the DBIF in the usual fashion of ALIAS modules. Relations opened via the DBIF have the proper scenario key field values for the current scenario placed in the cursen array of the /scenar/ block; these values are then used to construct keys for searches.

12.3.7 User Inputs

User inputs during the editing phase are in two forms: commands and assignments. The commands must be from the list

Figure 12-13. Sample Job Description Records From Ncjdat.descj

```
$LINE SCENARIO     CLASS       NCJOBT YARD      JSTYP  COMNUM CMETHD CUSTOMER
COMPLEXGRP DEFLT DAYSADDED APPROP_AWD AWD_ST ST_KL
KL_LN LN_DL DL_COM TIMUNT DATASOURCE   DATADATE ENTRY_DATE ENTRY_BY

   44 DEMO         LSD-41      NEWCON ANY       ORDFOL       1 MODULZ USN
11/01         0           1     12     6
   20     16      1 MONTHS 908          8/01/1984  8/02/1984 DBA
   45 DEMO         LSD-49      NEWCON ANY       ORDFOL       1 MODULZ USN
11/01         0           1     12     8
   23     24      1 MONTHS 908          8/01/1984  8/02/1984 DBA
   89 DEMO         LSD-49      NEWCON ANY       LEAD         1 MODULZ USN
11/01         0           1     12     8
   23     24      1 MONTHS 908          8/01/1984  8/02/1984 DBA
```

given in Figure 12-4. The assignments are display-page lines in the general form shown on the display page: the user enters a class name and job type character code, and numbers of ships in each period for that class-job. See Section 3 of the User's Guide for examples of the formats of prompts and responses.

12.4 DATA STRUCTURES

The emphasis in this section will be on data structures internal to an assigner run, though many of the input and output structures discussed in the previous section can be fruitfully thought of as assigner data structures. Examples are the schedule and job description relations, and the parameter and list menus.

The internal data structures will be discussed according to function rather than type. As a preliminary, Table 12-1 presents an annotated listing of the include files used by the assigner. The common blocks in these files range over all the ALIAS block subtypes: ordinary common blocks, record structures, and linked lists.

12.4.1 System Status Data Structure

In some sense the most important include file is the asgn.incl file, which contains four common blocks holding most of the system status information maintained during the initialization and editing phases. The values in two of these blocks, /casgn/ and /nasgn/, are continually stored into the cmnasn file so that system status information is recoverable in the event of an abort. These blocks contain such data items as the names of shipyards for which assignments are loaded, the arrays with row and column assignment totals, etc.

Cmnasn is created along with bufasn during initialization in the log-on group.

TABLE 12-1. Include Files Used By the Assigner

FILENAME	PURPOSE
ASGN	This voluminous include file contains four common blocks and several FORTRAN PARAMETER statements. It is the most important in-memory data structure for the initialization and editing phases of execution. In addition to flags, operating variables, display page totaling arrays, unit numbers, etc., the file contains the record structure (block /basgn/) which is used in communicating with the bufasn file.
ASHLDR	A record transfer structure used during the DB update phase to move assignments records (in their bufasn form) between the process data stack and an extra data segment used as a holding area. All assignments records in the complexity group currently being processed are maintained in the "hldbuf" data segment.
ASJD	ASsigner Job Descriptions. Used during the DB update phase to hold all the job description records for a particular class-job retrieved from the ncjdat.descj relation. There can be several such records, e.g. one for a LEAD job series type, one for an ORDFOL series type, etc. /ASJC/ is NOT a record structure (i.e. it is not used in the retrievals from the relation), but rather is a storage area consulted by the ncjodat record construction logic as necessary.
ASNOCR	ASsigner Outbound CuRsors. A common block of integer variables in which cursor indexes returned by the DBIF during the DB update relation-opening phase are stored. Used only by the update phase.
ASNVLD	The lists of valid (of-interest) ship class names, yard names, and job type code names as read from the assigner's three Command System list menus. Also, a list of valid job series type character codes for insertion in display page cells to indicate things like "lead ship in this period (L)". Since both job type and job series type are specified on the screen using single-character codes, this block has matching arrays of names and character codes to facilitate searching and retrieval.
ASOPRM	ASsigner Outbound PaRaMeters. FORTRAN parameter statements and variables with Command System parameter menu parameter settings of interest to the DB update phase.

TABLE 12-1. Include Files Used By the Assigner

FILENAME	PURPOSE
ASRBUF	ASsigner ship Record BUFfer. Holds the first form of schedule records generated during the DB update phase. Managed as a linked list which holds records only for the current complexity group. The records consist only of class name, job series type, the "display date" (the schedule date derived from the given job's column position on the display page), and the adjust date (the milestone used as the basis date during date spreading---need not be the same as the display milestone). Schedule records need only contain this information through the "hard-wire" tuple removal phase of the update process.
ASTFR	ASsigner Tuple File Record buffer. This include file contains two common blocks which function as record buffers for use in RELATE queries of and updates to the ncjodat.projj relation, and for similar operations performed on the schedule record holding file. During the actual relation update step it is necessary to have a record from each source current in memory at all times.
PVALUE	The System Core data structure which holds the current values for all command system parameter menus. Consulted during both the initialization phase and the DB update phase. In both cases the values are read by a service routine located in recomp.src and moved into common blocks dedicated to the assigner.
CONST	A block of commonly used constant values, e.g. the largest 32-bit integer number.
FLD05	Field list for reads and updates of the ncjodat.projj relation. The astfr include file must appear above it in any routine in which it is used.
FLD06	Field list for reads of the ncjdat.descj job description relation. Associated with the rcrd06 include file.
IOC	The standard ALIAS common block of key FORTRAN i/o unit numbers, e.g. those for terminal input and output.

TABLE 12-1. Include Files Used By the Assigner

FILENAME	PURPOSE
LPRNTS	The ALIAS array of logical variables (switches) controlling the operation of diagnostic prints.
PRMCRS	Permanently open ALIAS cursors (in each process). Used only by the iniprc and liston routines.
RCRD06	Record buffer which receives tuples from the ncjdat.descj relation. Used only as word-aligned temporary storage---the job descriptions are always moved into /asjd/.
SCENAR	Information about the current scenario and about cursors opened through the DBIF. Initial values for the block are swapped into the assigner process memory during the call to iniprc.
SCRCHR	FORTRAN parameter statements defining the command characters the assigner will recognize and a code number for each one. This block is actually used by several modules, so the assigner does not have a function it will perform for each command character found here.
SENPRM	Scenario system parameters. Required by the /scenar/ block.
SNRREF	A block of declarations supporting the scenario system's low-level utilities which search the scenario system extra data segment for scenario key values. Direct use of these utilities is made in the assigner when SELECTs are given.
TDDATE	The file of declarations and statement functions which supports full use of the ALIAS date manipulation utility subsystem by a routine.

During the DB update phase two additional common blocks hold status information, /asoprm/ and /asnocr/.

12.4.2 Valid Names Data Structure

The /asnvld/ common block contains lists of all the of-interest (valid) yard, class, and job type names turned "on" by the user in the assigner's Command System list menus. Job series type names are also read from the jstyp.legals relation and stored here.

The job type and job series type lists are maintained in dual form: the names are needed during initialization and DB update phases for comparison with field values in relations, but the single-chararacter code values used on the display screen to represent these names are needed during the editing phase. The names are maintained in an array, while the code values are maintained in corresponding elements of character*1 arrays. A match on an element of one array thus automatically yields an index number for the element of the corresponding array.

The code characters are read from the jobtyp.legals and jstyp.legals relations. Additional job type code names and series type names must be assigned unique character codes in these relations when they are added to the system.

12.4.3 Assignments Record Data Structure

A third block in asgn.incl (/basgn/) is the transfer record used to communicate with the bufasn file. Only a single assignment record is ever in memory at any given time, and it is stored in this block. Bufasn is a direct-access binary file with one record for each assignments record (class-job within a yard) displayable on the screen. Bufasn records are managed as a set of linked lists, one list per yard. The firstp array in /nasgn/ holds the record number of the first assignments record for each yard; subsequent records are pointed to using the first word of the bufasn record. The remainder of the record contains the class name, a storage location for ASNORDER (which holds a time stamp of when the record was first entered), the row-total of assignments in the record, and two arrays giving the number of assignments in each cell (display column) and a code indicating the character code to appear in each cell.

This data structure conserves on memory to the maximum extent possible, is efficient in terms of retrieval time during display generation, and provides abort protection since all assignments are maintained on disk at all times.

Display records are generated dynamically from this data structure during the refresh process, rather than being held in memory.

An additional assignments record structure is used during the DB update phase. This consists of the /ashldr/ common block, which is a record containing the equivalent of the two arrays from a bufasn record, and an extra data segment which can hold several of these records. The segment stores the records for all ships in the current complexity-group during the new-tuple generation process. The segment is used to conserve on memory; each assignments record requires 520 words, since 260 periods is the configured capacity of the assigner.

12.4.4 The Job Descriptions Data Structure

New tuples are produced by the DB update phase one class-job at a time after date-spreading has been completed. Several job description records may be required for any given class-job since each individual ship may be of a different job series type (e.g. LEAD, ORDFOL). All the descriptions for a given class-job at a given yard are thus maintained in memory simultaneously during schedule generation (to avoid time-consuming multiple searches of ncjdat.descj) in the /asjd/ common block. This block is a series of arrays dimensioned by the maximum memory capacity for job descriptions. The routines which use the block's contents first call the asgpf routine to get an index to a 'row' in /asjd/, and then just supply this index in any assignment statements using /asjd/ variables as the source.

/asjd/ is loaded by reads from ncjdat.descj using /rcrd06/ as the transfer record. The aspftr service routine is called after each read to transfer the given description to a 'row' in /asjd/.

12.4.5 The Schedule Tuple Data Structure

Schedule tuples are handled differently by the initialization and DB update phases. During initialization, a subset of tuple fields are read into a six-tuple array locally static in the asndbi routine. This supports the ordered-retrieval algorithm discussed in Section 12.5.1.

Several schedule record data structures are used during the DB update phase. The algorithm first generates one record per assignment in the /asrbuf/ common block, which is managed as a linked list with a capacity of 200 ships. Records are placed in the block in order of ship adjustment milestone in order to support the date spreading algorithm.

After dates are spread complete tuple images are generated and placed in a temporary holding file in the log-on group called tupfil. The /astfr/ block in the astfr.incl include file is the record structure used during read/writes from/to this file.

During the actual update of ncjodat.projj, records are read simultaneously from tupfil (into /astfr/) and from ncjodat.projj (into /astup/ in the astfr.incl file) and compared, with the /astfr/ image eventually being written into ncjodat.

12.4.6 Command Processing Data Structure

A command given during the editing phase may consist of up to three parts: its first or main-command character, its second or subcommand character, and one or more numbers separated by commas. The main and subcommand characters are compared with the list of known command codes in the scrchr.incl file of FORTRAN parameter statements and converted into index numbers. These indexes and the user-supplied numbers are then stored in variables declared in the asgn program unit (thus effectively global variables), and are passed to the proper main-command processing routine.

The search for a command character match is done on the scrchr parameter; the location of the match serves as the index. Note that the index values formally assigned to the characters in the rest of the scrchr.incl file therefore depend on the position of the character in the scrchr string.

12.5 PROCESSING LOGIC

The assigner is a very large program replete with logic and algorithms. In this section only the major algorithms whose structure and operation are not fairly obvious in the code will be discussed. To thoroughly understand how part or all of the program works it is necessary to consult the in-line documentation and the code itself.

Table 12-2 contains an annotated list of all of the FORTRAN routines in the assigner, not including utilities.

The calling tree diagrams which appear below show only those non-utility routines maintained as part of the ASGNxxx.src source code libraries. In addition, middle-level routines' subsidiary trees are typically shown on only the first diagram in which they appear in order to save space.

12.5.1 Initialization Phase Logic

Figure 12-14 is a calling tree diagram for the initialization phase of assigner execution. Also consult Figure 12-2 (initialization structure) for a pictorial display of the logic.

Table 12-2. Annotated List of Assigner Routines

ROUTINE	PURPOSE
ASCDAY	Low-level DB update logic utility which computes schedule milestone dates given a basis date and the index of a job-description in /asjd/. The job description gives time intervals between milestones; ascday just determines the appropriate number of intervals to increment the basis date by. An integer*4 function.
ASCDSP	A DB update schedule data modification routine which ensures that the award date in a schedule conforms to the default award day (DFLTAWDAY) field value in the appropriate job description relation. The routine is operative only when the display periods are years. An integer*4 function.
ASCMPG	DB update routine, "ASsigner-CoMPlexity Group." Finds out what complexity group each class-job of interest in a given shipyard falls in. Output is a series of code numbers corresponding to class-job names, with each group having the same code number (the numbers are arbitrary).
ASDWRN	Prints a warning to the effect that a hard-wire (no-assigner-modification) schedule has been deleted because there was no assignment left for it on the assigner display page. Part of the DB update logic.
ASGN	The main program unit for the assigner module. Both supervises the three main phases of execution by making appropriate calls and is the executive for the editing phase.
ASGNXT	Moves all the assignments records for class-jobs in a given complexity group into the hldbuf data structure (extra data segment) in preparation for date spreading and schedule generation. Part of the DB update logic.
ASGPF	Integer function returning the index in /asjd/ of the job description most appropriate for a given class-job in a given yard of a given series type. Presumes that a description for the class-job is available; this routine's task is to find the closest match on series type. Asgpf assumes an appropriate call to aspfld has been made to load the job descriptions for the given yard and class-job into

Table 12-2. Annotated List of Assigner Routines

ROUTINE	PURPOSE
	/asjd/.
ASHARD	DB update routine which removes ship records "associated" with "hard-wired" or no-assigner-modify tuples in ncjodat.projj. The method is to read ncjodat (via a selection) and, for every hard-wire tuple found, locate the ship record of the same class/job type with the closest display date, and remove it from the linked list in /asrbuf/. If there is no ship record in the same period, it is assumed the user wants the hard-wire tuple deleted, which is done.
ASHTRB	DB update routine which converts the hldbuf representation of assignments records for a given ship-job complexity group to the /asrbuf/ linked list of ship records. The ship records include both a display date and an adjustment date estimate; these are used by the date spreading logic.
ASJOI	Part of the DB update logic. Logical function which decides whether a given class-job needs to have schedule records generated for posting to ncjodat.projj. "ASsigner Jobs Of Interest."
ASN1ST	An assigner version of the fddate date utility, which returns the first day of a given period. An integer function.
ASNADD	Contains entry point asnins. Adds a new yard and/or class-job assignment to the assignments record buffer and the display page. Prompts the user for names and assignments, does error checking, and puts the response into the data structures. Implements the "A" and "I" commands.
ASNALO	Low-level editing phase routine which allocates a new assignment record (bufasn record) onto the free chain (linked list) of such records.
ASNALT	Implements the "M" command at a low level by writing the modified assignments record as given by the user to the record buffer.
ASNAMM	Implements the yard/class-job name changing capability. Prompt the user for the new name and

Table 12-2. Annotated List of Assigner Routines

ROUTINE	PURPOSE
	saves it into the data structure.
ASNCAL	Sets flags such that the next screen refresh will center around the yard whose index is given in the argument.
ASNCHD	Obsolete.
ASNCHK	Workhorse routine for the help subsystem; when the user (typically a developer) asks for the diagnostic support part of the help subsystem (via ??) and gives a command there, this routine executes the command.
ASNCLN	Removes assignments based on a load from the relations with schedules for the historical and current epochs by re-reading those relations and <u>decrementing</u> the assignments buffer (bufasn) for every one found. Clumsy and time-consuming but the only way short of marking each bufasn element, also clumsy.
ASNCLR	Does a screen clear or formfeed, depending on whether output is to the screen or the printer.
ASNCMD	This routine prompts for, reads, and decodes user command input during the editing phase. Commands are broken into the main and sub command characters and any numeric specifications which follow them.
ASNCNV	Executive which supervises conversion of the bufasn assignments records into ncjodat.projj tuples. See Figure 12-8 for a summary of its flow of control.
ASNCOD	Servant of asndbi, used to set the job series type letter code value for a particular cell of the buffer. Ensures that the code shown on the page end up being the one attached to the "highest" ship "in" the cell, where the order from highest to lowest is, e.g., lead ship, first follow, lead in yard, ordinary follow. If there are 8 ships in the cell (i.e. ad 8 is displayed there on the screen) and two of them are a lead ship and a first follow, the cell will show an 'L'.
ASNCPY	Makes a new copy of a yard or a class-job within a yard, prompting user for the names for the new copy,

Table 12-2. Annotated List of Assigner Routines

ROUTINE	PURPOSE
	and making the necessary changes to bufasn and /asgn/.
ASNDBI	Executive for the load of schedules from the data base and their conversion into assignments records in bufasn. Reads records in key order from all six schedule relations simultaneously (see text), constructing assignment records as it goes. Creates new yards and classes within yards as necessary. On completion of the DB read, reorders assignment records according to input order if user has asked for that.
ASNDBR	The DB update phase routine which actually posts the newly created schedules to the ncjodat.projj relation. Must ensure that at the close of the assigner session the schedules in ncjodat are completely in consonance with the assignments which were showing on the display screen. Must also take into account the fact that the user could have been working with a limited period of time or a limited list of valid yards/classes/job types. Operates in a fashion basically similar to the asndbi routine: a tuple from the relation and a record from the new tuple holding file are always kept constant in buffers; action decisions are made on the basis of a comparison of their key values. The actions possible are to get the next tuple, update the existing one, delete it, or add a new tuple.
ASNDEL	Deletes one or more assignments records from the data structure, up to an entire yard. Implements the "D" command.
ASNDOT	Mid-level utility which prints the prompt ("dots") for assignment record addition or modification. Also reads, checks, and decodes the input.
ASNDWN	Does the computations for a next-vertical-page command.
ASNEC	Reports a command input error or some other status condition to the user, pausing to let the user read the message.
ASNEND	An obsolete close-relations and finish up routine.

Table 12-2. Annotated List of Assigner Routines

ROUTINE	PURPOSE
ASNEOI	Resets i/o unit numbers back to the terminal when an end-of-file is encountered on the current input unit number. Useful for detecting and resetting after the end of execution of the "P" command, for example.
ASNFND	Low-level routine which locates a particular class-job's assignments in a particular yard and brings it into the assignment buffer in /asgn/.
ASNFOL	Implements the vertical page-forward command (+ and ++). Causes the index numbers of the top yard and class-job (used by the display refresh logic) to be recomputed.
ASNGOQ	Logical function which asnout uses to ask the user if he wants to skip the DB update step.
ASNHLD	Implements the "H" (hold) command; suspends the assigner process and reactivates the Core process.
ASNHLP	Assigner help subsystem executive. Responds to the "?" command. Prompts the user with a menu of help choices. Accepts and implements the response.
ASNHUL	Responsible for updating the hull numbers in schedules newly posted to ncjodat.projj during the DB update phase. As posted the schedules have negative hull numbers to ensure that no unary key violations occur as a result of collisions with "hard-wired" schedules. The routine carries out its task by having RELATE execute the newhul.rprocs EXECUTE file, which contains the actual logic. However, asnhul must write proper scenario key field values into newhul.rprocs before it is executed so that scenario security is maintained.
ASNINI	Executive for the initialization phase. Does or supervises completion of everything necessary before user assignments editing can begin. Major steps include reading the iniasn.sysro configuration file, creation of the bufasn and cmnasn working files, load of valid lists from the list type relations, and loading of schedules from the data base and conversion of these into assignments records in bufasn.

Table 12-2. Annotated List of Assigner Routines

ROUTINE	PURPOSE
ASNINS	An entry point in asnadd which lets the user specify where on the display page the new yard/assignments are to be placed.
ASNLBL	Prints the top two rows of the screen display, which give status information (i.e. name of the current scenario) and the period labels.
ASNLBS	Performs part of the initialization of the /asgn/ common block, in particular for those variables whose values depend on setting in the Command System's assigner parameter menu, e.g. time units/period type option. Formats and stores relvant parts of the screen display.
ASNLEV	Closes all files and relations at the end of the DB update phase, preparatory to assigner process termination.
ASNLFT	Contains the asnrgt entry point as well. Implements the page-right and page-left horizontal (over-periods) paging commands (>, >>, < <<). Recomputes column index specifications used by the display refresh logic.
ASNLPR	Implements the "P" command by redirecting display output to the user's default hard copy output device and by sending all available pages to this unit.
ASNMNP	Part of the assigner help subsystem; displays the values of selected Command System assigner parameter menu parameters for user inspection/reminder.
ASNMOD	Modifies an existing assignment record (i.e., implements the "M" command). Prompts the user with the existing assignments line and the dots and updates the data structures.
ASNMOV	Implements the "Move option of the "Relocate" command; moves a yard or a class-job's assignments to a different location on the display screen or to a different yard.
ASNOUI	Initialization routine for the DB update phase logic. Open relations and the tuple holding file and sets up

Table 12-2. Annotated List of Assigner Routines

ROUTINE	PURPOSE
	important variables based on Command System parameter menu settings.
ASNOUT	Executive for the DB update phase of execution (the "outbound leg"). See Figure 12-7.
ASNPOP	Implements the "^" and "Q" commands; an interface routine between the editing and DB update code which calls the DB update executive.
ASNPRN	Formats and prints an assignments buffer record, i.e. part of the contents of the valasn array in /asgn/.
ASNPRO	Conditionally prints prompt text, based on the setting of the prompt flag in /asgn/. Prompt is set to true when operation is interactive, false when responses are taken from a file or some other source. Routines which prompt through asnpro are thus appropriate for use as processing utilities as well as for user interaction.
ASNPRV	Implements the vertical page-up command ("- and --"). Computes the index numbers of the new top yard and class and stores them for reference by the display-refresh logic.
ASNRDC	A sophisticated terminal prompt-and-read utility. Takes prompt text and directives and returns the user response. Optionally takes response as input also and just runs it through its check logic. Checks for pop (undo) character and for help requests, and prints help from the hlpasn file if a "?" is given. Upper- or lowercases the input.
ASNREF	Conditionally calls asnrfh for a screen refresh: does so if prompt is true (we're interactive) and if the user has requested auto-refresh.
ASNREO	Re-orders the display order of assignments within a yard according the their input order, as obtained from the values in the ASNORDER field of the schedule relations. Called only when user has chosen INPUT ORDER rather than ALPHABETIC on the parameter menu.
ASNRFH	Performs a screen refresh, i.e. prints the current display page of the assignments buffer to the screen.

Table 12-2. Annotated List of Assigner Routines

ROUTINE	PURPOSE
	A mid-level executive which does lots of retrievals.
ASNRGT	Page-right. An entry point in asnlft; see its description above.
ASNRLC	Implements the relocate ("R") command and its permutations. Either repositions a yard or class-job on the display page or make a copy of a class-job assignment record under another class-job name.
ASNSEE	Executive for the "??" (diagnostic assistance) help option.
ASNSWP	Inoperative.
ASNTPE ASNTPI ASNTPX	These three entry points of the asntpx routine form the schedule-tuple retrieval subsystem serving asndbi during the initialization phase. The routines manage a static (local) buffer which holds six schedule tuples, one per relation. The tuple images in this buffer are the next valid tuple in sequence from each relation. Asntpi loads this buffer. A call to asntpx returns the image which has the lowest key value of the six available; asntpx fills in the "empty" location with a new image before it returns. Asntpe just closes the schedule relations when the read is complete.
ASNTUP	A DB update utility routine used by asndbr to retrieve the next tuple from ncjodat.projj into the tuple holding buffer. Checks for both actual end-of-file and for end-of-scenario.
ASNUNL	Takes a schedule relation tuple image and unloads its fields into individual variables, passing their values back.
ASNVAL	Called when an old bufasn/cmnasn exists and user wants to use it. Flushes yards and classes which are not valid under the current invocation (i.e. not turned on in the list menus, or not even appearing if this is a different scenario).
ASNWID	Given a start period for screen display (i.e. the period number of the leftmost column, returns the number of columns to print and the index of the last

Table 12-2. Annotated List of Assigner Routines

ROUTINE	PURPOSE
	column. A utility.
ASNYRD	Used during display generation, prints a yard name and the "--+--+--" grid lines.
ASOCMP	A character string comparision utility used by asndbr to decide if a given schedule key value is greater or less than another. An integer function returning -1 (key less), 0 (key equal), or +1 (key greater).
ASODEL	A servant of asndbr, called when asndbr thinks it has an ncjodat.projj tuple requiring deletion. This routine decides if the deletion is appropriate (might not be a valid job for this invocation, might be a no-assigner-modify tuple already processed by ashard) and does it if necessary. Asubdl is called to delete tuples in subsidiary relations.
ASPFLD	Reads the job description tuples for a given class-job in a given yard into the /asjd/ storage block for use by other DB update routines.
ASPFTR	A slave of aspfld which just copies a job description from the /rcrd06/ buffer in which RELATE placed it into an index location in /asjd/. A simple xmit is not feasible due to the structure of /asjd/, which is in turn mandated by the requirements of the data calculation logic.
ASPRD2 ASPRED	These two routines implement the schedule date-spreading logic of the DB update phase. They operate on the /asrbuf/ linked list of ship records, changing only the adjustment-basis dates. Asprd2 was the original algorithm; it is not in the calling tree, having been replaced by the modified version now called aspred, but is functional. It was replaced as a matter of taste and might be offered as a parametrically invoked option in the future.
ASTUPF	Part of the DB update phase, astupf converts /asrbuf/ ship record to tuple images in the tupfil direct access holding file. It follows date spreading and preceeds the actual update of ncjodat.projj.
ASUBDL	When a no-assigner-modify (hard-wired) tuple is deleted from the ncjodat.projj relation by the

Table 12-2. Annotated List of Assigner Routines

ROUTINE	PURPOSE
	assigner (might happen when its whole yard was deleted, for example) then any tuples dependent on it in subsidiary relations must also be deleted, much as the DBU deletes subsidiaries automatically (this is not required when assigner-modifiable tuples are deleted because it is assumed that the DBU marks any "father" schedules as AUTOMOD="NO" when son tuples are added in the subsidiary relations). This routine does the extra deletions, learning which relations have something to be deleted by the status of the bit map in the SUBRELUMAP field of the schedule tuple about to be deleted (the DBU is assumed to maintain this field as well).
ASYCLS	Part of the DB update logic. Constructs a sorted list of the class-jobs for which update must be done in a given yard. Note that repair jobs can be ignored.
CKPF	Checks to see if a job description is available when the user adds or modifies assignments, thus providing advance warning of the necessity to go add the description using the DBU during or before the DB update phase in cases where no description has been entered. Logical function.
CMNGET CMNSAV	These two entry points in the cmnget routine retrieve and save /asgn/ status variables from/to the cmnasn file.
DBASIS	A character function used during the initialization phase to set the name of the KEEL/DRYDOCK field in field lists for schedule relation reads depending on whether the relation holds repair or new construction job data.
GETASN	This routine and its putasn entry point save and return records from the bufasn assignment record holding file.
INICLS	Does the necessary setup to establish a new class in a yard. Doesn't create the bufasn record, just sets /asgn/ values.
INIYRD	Does the necessary setup to establish a new yard.

Table 12-2. Annotated List of Assigner Routines

ROUTINE	PURPOSE
LOCYRD	Does a binary search of the (sorted) array of existing yard names for a match, returning the element location of the match. An integer function.
NEWYRD	Adds a new yard to the list for which assignments can be made, prompting the user for the name. Ensures that the list of yards remains sorted.
NXTCLS	Retrieves the next tuple of interest from the schedule relation open on the given cursor. Reads records (points under some circumstances) until one matching all retrieval criteria (valid display basis date, yard name, class name, etc. are found).
NXTCLZ	A debugging support routine which prints a tuple and other data to ioutp. Used mainly by nxtcls.
PUTASN	An entry point in getasn, saves an assignments record to the given location in the bufasn file.
REMCLS	Removes a class (i.e. an assignment record) from a yard completely. Pulls it out of the bufasn holding file linked list and makes the necessary /asgn/ changes.
REMYRD	Removes an entire yard and all its assignments records.
RESTAT	Entry point in svstat; see below.
SCNGET	An entry point in CMNGET which read the first record of an existing cmnasn file to see what scenario it was created under.
SVSTAT	This routine and its entry point restat save and restore the current values of assigner control variables before and after the "P" command is executed. "P" is implemented by using the standard logic but with alternative control settings; thus the setting must be kept and restored if the user is to be left in the same state as before "P" was given.
TMSTMP	An integer*4 function which computes a time-stamp for placement in the ASNORDER field of newly generated schedule records so that on next initialization the assigner will be able to retrieve them in the order

Table 12-2. Annotated List of Assigner Routines

ROUTINE	PURPOSE
	they appeared on the display. Every member of a given class-job gets the same time stamp.
TUPFRD	A low level routine used by asndbr to fetch the next schedule record from the tuple holding file.
VALCLS	Given a class name (including job type definition character, returns whether it is valid (usable) under this scenario, i.e. whether class and job type are "on" in list menus. A logical function.
VALYRD	Logical function returning .true. if the given yard name is on the valid list for this scenario and assigner invocation.
VLDLST	Initializes the /asnvld/ common block's lists of yards, classes, and job types that are valid for this invocation, i.e. that it's ok to work with. Does this by calls to the liston and qsortc utilities, primarily.
VLDLSZ	A diagnostic utility for vldlst which prints out the lists of valid names and codes after they're set up.
YCASN YCASNI YCASNR	Service routines which implement a "control-Y" capability to abort printout of unwanted screen refreshes. Ycasn is called when the user invokes the interrupt by pressing the control and Y keys, ycasnr can be called later to detect that the interrupt occurred, and ycasni resets the flag which remembers the interrupt. The routines do not issue the ON statement; the routine(s) using the capability must issue the ON.

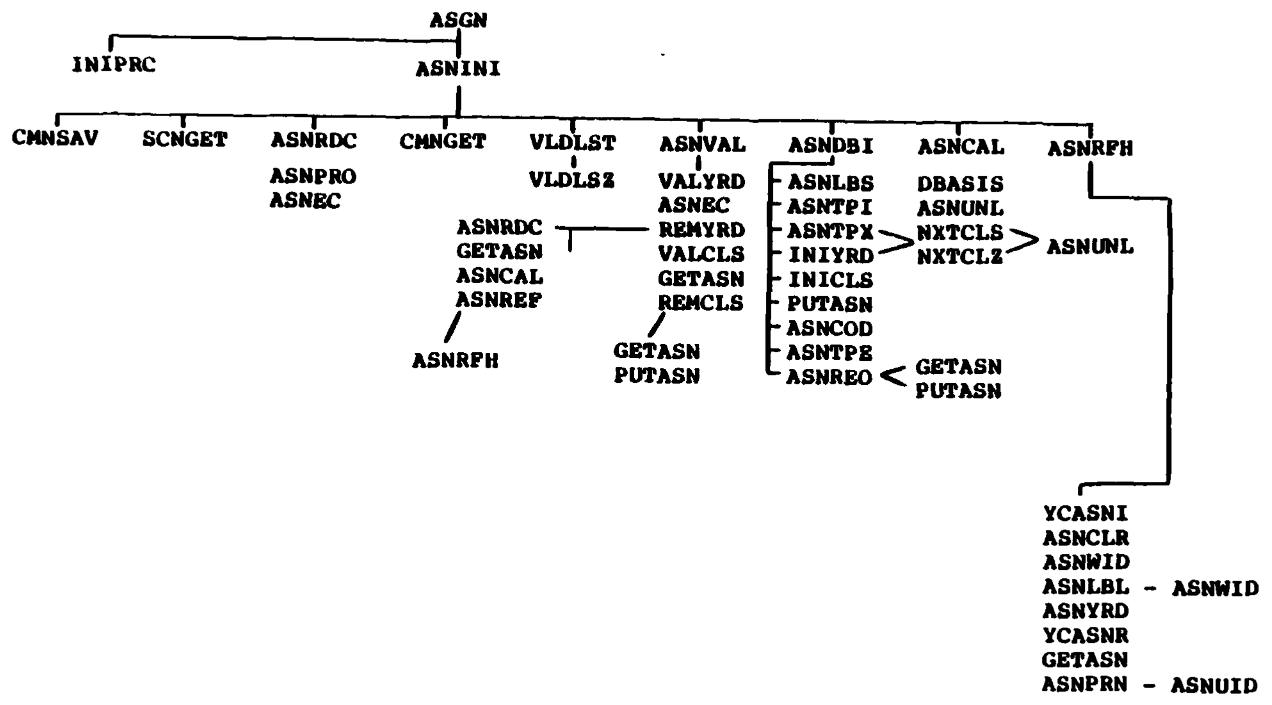


Figure 12-14. Assigner Initialization Phase Calling Tree

Initialization is overseen almost entirely by the ASNINI routine. The operation of this routine is fairly straightforward: it reads the iniasn.sysro files, opens and/or creates other files which will be accessed via FORTRAN i/o, sets up the lists of valid class, yard, and job type names via a call to vldlst (which in turn just uses the liston utility), and reads the schedule relations to construct assignment records. At the close of initialization a refresh is done (by a call to asnrfh) in order to present the first screen page.

The non-obvious parts of the logic have to do with the schedule read, conducted by asndbi, and with what happens when there is an existing assignments buffer (bufasn file).

This latter condition will occur only when the last user to execute the assigner in the log-on group aborted during the editing or DB update phases. The user must be prompted for a desire to recover from the abort rather than starting fresh. If recovery is desired, then no DB read is required; the system is returned to its state at the time of the abort by reading the contents of the cmnasn file into memory and using the contents of the bufasn file.

There is a catch, however, embodied in the call to asnval. Since the /asnvld/ common block is not saved on disk as is /asgn/, the valid name lists must be re-initialized via a call to vldlst after an abort. However, there is no guarantee that the user has not changed the settings in the valid lists since the abort. It is therefore possible that bufasn may contain assignments records with invalid names. These are flushed by asnval so everything is consistent.

The read of data base schedules, conducted by asndbi, centers around use of asntpx and its subsidiaries to retrieve schedules from the relations. In order to make initialization efficient, it was desirable to have the schedule tuples be read in order of yard and class-job, so that assignments records could be constructed one at a time with no necessity to go back and work on them again. At first blush this seems no problem, since one can just read from the relations on an index consisting of SCENARIO,YARD,CLASS,(NC RE)JOBT. However, up to six relations are involved, each open on a separate cursor, and a typical assignments record might have schedules resident in several relations.

It is thus necessary to construct a single virtual relation. Unlike the horizontal construction effected by a SELECT, this needs to be a vertical construction in which, apparently, the contents of all six relations are copied into a single temporary file which is then indexed by the given fields and read sequentially.

The tremendous inefficiency involved in creation of a temporary file is avoided by the following algorithm:

1) Open the relations and retrieve the first tuple from each for the given scenario on the appropriate index into a holding buffer (thus a buffer with a six-tuple capacity). This is done by asndbi.

2) Have the main read routine (asndbi) call a utility (asntpx) which does an in-memory sort of the six tuples in the buffer according to the given keys, returning the one with the lowest key value.

3) This utility in turn calls a service routine (nxtcls) to get the next tuple from the relation whose tuple was just selected, placing it in the 'vacated' location of the holding buffer. Nxtcls also ensures that the tuple is still for the given scenario, is for a valid class, yard, job type, etc., and that it is not an earlier-datadate-representation of the tuple just used. Nxtcls places a high ascii-collating sequence character in the most-significant key location of the buffer when it encounters end-of-file or end-of-scenario in a given relation, thus ensuring that the given buffer location will not emerge at the top of the in-memory sort.

This algorithm is extremely efficient and produces the desired read-ordered behavior. It employs record reads on the relations in preference to record points, doing points only when it finds it has read into a new yard (it does one point for each valid yard name).

When the algorithm detects that a given assignment record is complete (by asntpx returning a schedule for a different class-job), the given record is processed into the /asgn/-bufasn data structure just as though the user had entered it interactively.

At the close of DB reading the assignments records are reordered according to ASNORDER field value (stored in bufasn record) if the user has specified the INPUT ORDER parameter option for class-job ordering (this is done by asnreo).

12.5.2 Editing Phase Logic

Figure 12-15 presents the calling tree diagram for the editing phase. In spite of the very large number of routines mentioned on the diagram, the logic of the editing phase is fairly straightforward.

The logic is organized around response to specific user commands, with command prompting and response overseen by the asgn program unit. With the exception of the asncmd command retrieval utility, every other routine called by asgn is an executive for the processing of a particular command.

Note that asnec, called almost everywhere, is an error-reporting utility.

The paging and display-generation algorithm can be somewhat obscure because it is highly data-driven and is distributed among several routines. When the user requests a page up/down or right/left, the only processing involved is recalculation of the

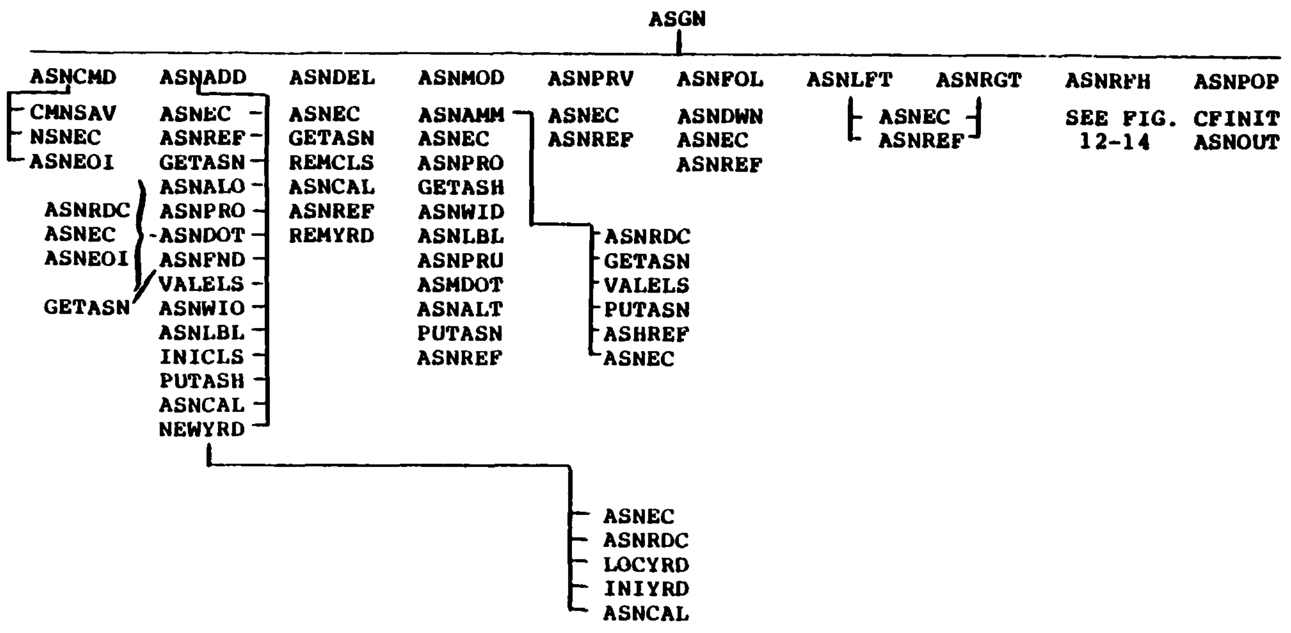


Figure 12-15. Assigner Editing Phase Calling Tree Diagram

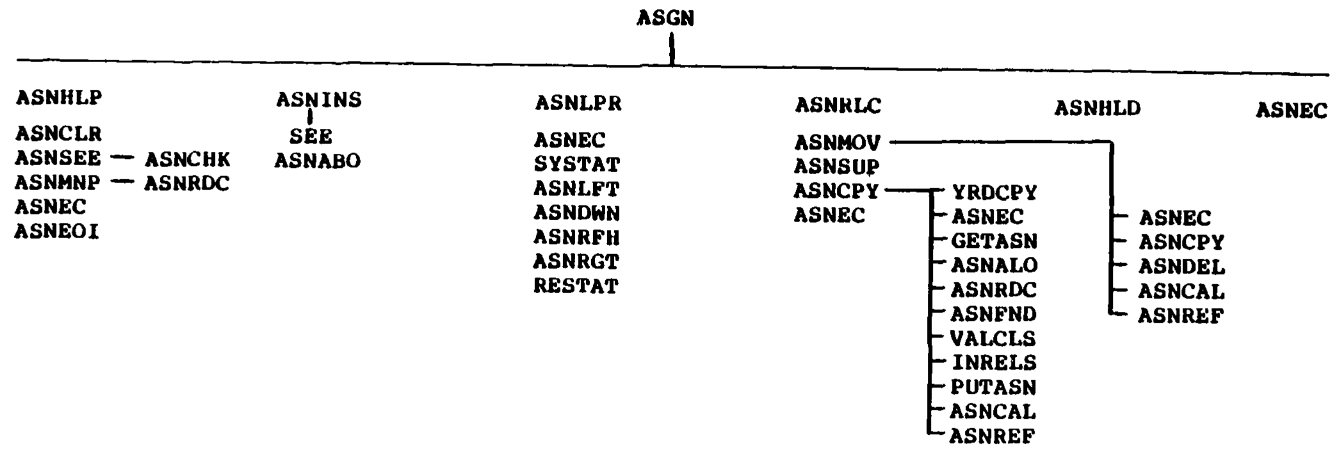


Figure 12-15. Assigner Editing Phase Calling Tree Diagram (Continued)

index numbers of the topmost yard/class-job and the leftmost column to show at the next refresh. These computations are complicated by a desire not to have yard names "hanging" at the bottom of the display with their class-jobs all appearing on the next page.

The refresh logic (asnrfh) takes these index settings as input. It prints the first few lines of the display, the header, and then retrieves bufasn records as necessary to print the assignments records. It must pay attention to proper placement of the grid lines which make it easier to read the columns, and to placement of row and column totals.

Note that most system calls for a refresh are done through the asnref routine, which consults the setting of the AUTO REFRESH parameter and just returns if the user does not want an automatic refresh after processing of each command.

Processing of the assignment-modification commands is more concentrated and linear, typically involving error checking to ersure the user has asked for something sensible, a prompt for the new assignment or modification, more error checking on the response, and posting of the result to bufasn.

12.5.3 Data Base Update Phase Logic

Figure 12-16 is a calling tree diagram for the DB update phase. Note also that Figures 12-7 and 12-8 and Section 12.1.3 summarized the structure of the phase. This section will concentrate on subtle parts of the algorithm.

The basic idea of the update is to make the bufasn assignments record structure reflect only the assignments to be updated in ncjodat.projj, generate the corresponding schedule records and place them in the relation, give the schedules as realistic a set of hull numbers as possible, and clean up by getting rid of bufasn and cmnasn.

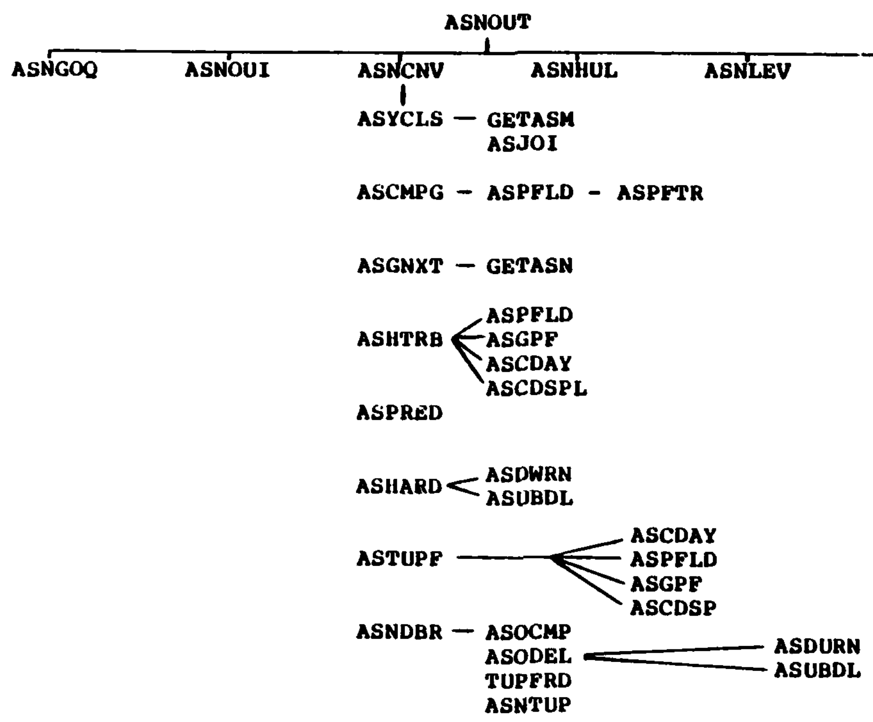


Figure 12-16. Assigner DB Update Phase Calling Tree Diagram

12.5.3.1 Bufasn Preprocessing

This brute-force step removes any assignments in bufasn which were placed there during initialization as a result of reads of schedule records from ncjodat.histj and ncjodat.currj. Assignments originating in the rejobt relations are no problem because all repair-type jobs are ignored during the update anyway (they must be ignored because there is no way for the user to specify in limited screen space exactly which ship (class-hull-comnum) a given repair job is to be done on, and this is a crucial piece of information about a repair job).

The removal is done by re-reading the historical and current relations, using a selection rather than the asntpx-based logic of the initialization phase. For each schedule returned the corresponding bufasn assignments record is found and the proper column decremented.

12.5.3.2 Schedule Creation

The schedule creation algorithm must perform four tasks: it must generate detailed information from the summary data provided by the assignments records, it must (optionally) spread jobs over time intelligently, it must be sure not to touch any hard-wired schedule tuples in ncjodat.projj, and it must ensure that the schedules it generates are placed in ncjodat efficiently and in such a way that the relation's contents accurately represent the user's expressed desires.

Generation of the detailed schedules is done by creating a set of schedule records of number equal to the grand total showing at the bottom right of the assignments display screen MINUS all historical/current jobs and all repair-type jobs. Each record is associated with a given yard name, class name, job type name, job series type name, and contains a display date which is the first day of the period represented by the display page column the assignment appeared in. This is the total amount of information supplied by the user.

The rest of the information comes from the job description relation ncjdat.descj, which is queried for matches on the scenario, yard, class, job type, and job series type fields.

The full schedule records are generated by a two-pass query of ncjdat for each yard and complexity group. The first query is used only to compute the adjustment date for each schedule (i.e. the milestone named in the ADJUST BASIS parameter by the user) based on the display date. Records in this limited-information form (in the /asrbuf/ linked list) are then processed through the date spreading algorithm.

The date spreader takes all the schedule records in a given complexity group (may be limited to a given class-job), finds the first date and the last date of the interval they span, computes an average interval between ships, and recomputes their adjustment dates such they are the average interval apart. The algorithm is constrained to produce adjustment dates that fall within a single display-period duration later than the original adjustment date, thus ensuring that when a new display date is computed from the new adjustment date the given assignment would still appear in the same display page column.

The output of the date spreader is then passed through the filter of the hard-wire tuple compensator ashard. This routine reads ncjodat.projj for tuples in the current scenario with AUTOMOD field values of "NO", finds which new schedule record most closely corresponds to each such tuple, and removes the schedule record from the /asrbuf/ linked list. Hard-wired tuples can thus be ignored during the later relation update pass, since they are already compensated for.

At this point /asrbuf/ holds a number of schedule records equal to the adds/updates which will be done on ncjodat.projj for the given yard and ship complexity group. Full-scale schedule tuples are generated by a second query of ncjdat.descj and by combining its information with that in /asrbuf/, with output to tupfil.

Asndbr then supervises the update of ncjodat.projj, using tupfil's contents. It reads tupfil and asndbr concurrently, performing either a skip, delete, or update for each ncjodat tuple and an add or update for each tupfil record. Hard-wired tuples and those not in a class, yard, job type, or period of interest are skipped. Those which match the keys of the current tupfil record are updated with that record. Those with smaller keys are deleted. When the ncjodat tuple keys are larger, tupfil records are added.

12.5.3.3 Hull Number Assignment

Figure 12-17 displays the text of the newhul.rprocs RELATE EXECUTE procedure file. This file is processed by the asnhul FORTRAN routine and written to a temporary with the proper scenario key field values replacing each instance of "IMAGINATION" in the original. It is then executed.

Its goal is to assign the most realistic hull number possible to each schedule in the current scenario (not only those on the valid lists for this run) given the state of the data base. It first reads the ncjodat.histj and ncjodat.currj relations to determine the maximum hull number in each class. It then assigns this hull number to the first ship of each corresponding class in ncjodat.projj, making the assignment in the PROGVAR1 working variable, however. There are usually some projected classes which do not appear in the current and historical relations; the procedure next attempts to extract a first-ship hull number from the class name itself, e.g. "51" from DDG-51. Where even this fails, the earliest-delivery ship of the class is assigned "1" as a hull number.

Figure 12-17. Text of NEWHUL.RPROCS Execute File

```
 1      NOTE 23
 2      NOTE 28
 3      NOTE 47,49,51,53,61,68,71,76,82,103,105
 4      NOTE            first lines of file must contain posn of lines
 5      NOTE            where alias will substitute in name of current scen
 6      NOTE
 7      NOTE    THIS RELATE PROCEDURE FILE IS TO BE CALLED BY
 8      NOTE    ALIAS MODULE ASSIGNER.
 9      NOTE
10      NOTE    ITS PURPOSE IS THE UPDATING OF SHIP HULL NUMBERS TO
11      NOTE    FORM A CONSISTENT, UNIQUE SERIES AFTER ASSIGNMENTS HAVE
12      NOTE    BEEN CHANGED
13      NOTE
14      OPEN FILE NCJODAT.PROJJ;MODE=SHARED;PATH=PROJX
15      OPEN FILE NCJODAT.CURRJ;MODE=SHARED;PATH=CURRX
16      OPEN FILE NCJODAT.HISTJ;MODE=SHARED;PATH=HISTX
17      NOTE
18      NOTE    GET MAX HULL NUMBERS FOR EACH CLASS
19      NOTE
20      SET PATH HISTX
21      SELECT SCENARIO,CLASS,HULL=$MAX(HULL BY SCENARIO,CLASS) &
22        UNIQUE BY SCENARIO,CLASS WHERE SCENARIO= &
         "IMAGINATION"
24      COPY TO HLTMP;ERASE;RETENTION=TEMPORARY
25      SET PATH CURRX
26      SELECT SCENARIO,CLASS,HULL=$MAX(HULL BY SCENARIO,CLASS) &
27        UNIQUE BY SCENARIO,CLASS WHERE SCENARIO= &
28       "IMAGINATION"
29      COPY TO HLTMP;RETENTION=TEMPORARY
30      NOTE
31      NOTE    INDEX MAX HULL NUMBERS; UNARY INDEX REQUIRED FOR
32      NOTE    LET COMMANDS GIVEN THROUGH A SELECTION
33      NOTE
34      OPEN FILE HLTMP;RETENTION=TEMPORARY
35      SELECT CLASS,HULL UNIQUE BY HULL WHERE HULL=$MAX(HULL BY CLASS)
36      COPY TO HLTMP2;ERASE;RETENTION=TEMPORARY
37      OPEN FILE HLTMP2;RETENTION=TEMPORARY
38      LET HULL=HULL+1
39      CREATE INDEX BY CLASS;UNARY
40      NOTE
41      NOTE    GET THE PROJ.   SCHEDULE RELATION SET TO MOVE THE MAX
        NOTE    HIST/CURR
42      NOTE    HULL NUMBERS BY CLASS IN
43      NOTE
44      SET PATH PROJX
45      SET INDEX SCENARIO,CLASS,DELIVERY
46      LET PROGVAR1=1 FOR SCENARIO=&
47       "IMAGINATION"
48      LET PROGVAR1=$LAST(PROGVAR1,CLASS) FOR SCENARIO=&
49       "IMAGINATION"
50      LET PROGVAR2=0 FOR SCENARIO=&
51       "IMAGINATION"
```

Figure 12-17. Text of NEWHUL.RPROCS Execute File

```
LET PROGVAR2=1 FOR PROGVAR1=0 AND SCENARIO=&
 "IMAGINATION"
NOTE
NOTE    MOVE MAX HIST/CURR HULL NUMBERS INTO FIRST TUPLE EACH CLASS
NOTE    WHERE NO MAX/CURR HULL NUMBER, SET TO NUMBER IN CLASS NAME
NOTE    IF THERE IS ONE
NOTE
SELECT PROJX.@,MHULL=HLTMP2.HULL WHERE PROJX.CLASS=HLTMP2.CLASS AND &
                                PROJX.PROGVAR2=1 AND PROJX.SCENARIO= &
 "IMAGINATION"
LET PROGVAR1=MHULL
SELECT
NOTE    PARSE FOR THE FIRST NUMERIC SUBSTRING OF THE CLASS FIELD
LET PROGVAR1=$SUBSTR(CLASS,$MATCH(CLASS,"[1-9]"), &
            $MATCH($SUBSTR(CLASS,$MATCH(CLASS,"[1-9]")), "[/0-9]")-1)&
            FOR PROGVAR1=0 AND SCENARIO= &
 "IMAGINATION"
NOTE    CATCH ANY LEFT AND SET THEM TO 1
LET PROGVAR1=1 FOR PROGVAR1=0 AND SCENARIO= &
 "IMAGINATION"
NOTE
NOTE    NOW SET HULLS FOR EACH CLASS IN INCREASING ORDER
NOTE
LET PROGVAR1=$RTOTAL(PROGVAR1,SCENARIO,CLASS) FOR SCENARIO = &
 "IMAGINATION"
NOTE
NOTE    NOW MAKE SURE THAT NO SHIPS FLAGGED BY THE SCHED EDITOR
NOTE    AS UNCHANGEABLE WILL HAVE THE SAME HULL AS AN UNFLAGGED SHIP
NOTE
IF PROJX.AUTOMOD="NO" AND PROJX.SCENARIO= &
 "IMAGINATION"
OPEN FILE NCJODAT.PROJJ;MODE=SHARED;PATH=PROJ2
SELECT LINE=PROJ2.$LINE,AJUNK=PROJ2.PROGVAR1,PROJX.PROGVAR1, &
 PROJX.HULL,PROJX.AUTOMOD &
  WHERE PROJ2.PROGVAR1=PROJX.HULL AND PROJ2.SCENARIO=PROJX.SCENARIO&
    AND PROJ2.CLASS=PROJX.CLASS AND PROJX.AUTOMOD="NO"  &
    AND PROJ2.COMNUM=PROJX.COMNUM
COPY TO HLTMP3;RET=TEMP
OPEN FILE HLTMP3;RETENTION=TEMP
CR IN BY LINE;U
SELECT PROJX.PROGVAR1,HLTMP3.LINE,FROM=HLTMP3.PROGVAR1 &
  WHERE PROJX.$LINE=HLTMP3.LINE
LET PROGVAR1=FROM
CLOSE PATH PROJ2
PURGE FILE HLTMP3
ENDIF
NOTE
NOTE    NOW TRANSFER THE HULL NUMBERS TO THE HULL FIELD
NOTE
SET PATH PROJX
LET HULL=-1000+$RTOTAL(-1) FOR AUTOMOD="YES" AND SCENARIO= &
 "IMAGINATION"
```

Figure 12-17. Text of NEWHUL.RPROCS Execute File

```
LET HULL=PROGVAR1 FOR AUTOMOD="YES" AND SCENARIO= &
 "IMAGINATION"
NOTE
NOTE    CLEAN UP
NOTE
CLOSE PATH PROJX
CLOSE PATH CURRX
CLOSE PATH HISTX
PURGE FILE HLTMP
PURGE FILE HLTMP2
**
```

The remaining ships of each class are then assigned hull numbers in increasing order starting with the first hull for that class, still in the PROGVAR1 variable. Next the hull numbers are revised, so that there are no unary-key conflicts with hard-wired tuples, by swapping the PROGVAR1 hull numbers of hard-wired tuples with those of tuples whose PROGVAR1 value matches the actual HULL number in the hard-wired tuple. Finally the HULL field can be assigned the value of the PROGVAR1 field for all non-hard-wired tuples.

12.6 FILES USED BY THE ASSIGNER

The assigner uses a large number of files and relations. The iniasn.sysro and hlpasn.sysro permanent files are consulted. Two permanent files named bufasn and cmnasn are created in the executing user's log-on group, and purged only on successful assigner process completion. Temporary files tupfil and extmpxyz are created as working areas during DB update execution.

The ncjodat.histj, ncjodat.currj, ncjodat.projj, rejodat.histj, rejodat.currj, rejodat.projj, ncjdat.descj, valcls.mnurel, valyds.mnurel, vljtyp.mnurel, jobtyp.legals, and jstyp.legals relations are all consulted; ncjodat.projj is altered.

Assigner source code is in the .src group, in files asgna, asgnan, asgnan, asgnanc, asgnand, asgnane, asgnani, asgnanm, asgnanr, asgnant, asgnao, asgnat, asgnc, asgnr, asgny, and recomp. Object code is in corresponding files in the .obj group. Linkable object code is in asgn.obj. Program files are in asgn.prog and tasgn.prog. The GLUE, LINK, and MAKE procedure files are in asgn.merge, asgn.link, and tasgn.link respectively.

12.7 INTERFACES

The assigner depends on the integrity of the data base in several important respects. First, it assumes that the yard, class, and job type names appearing in all schedule records are represented on the candidate lists in valyds.mnurel, valcls.mnurel, and valjtp.mnurel. This will be true as long as only the DBU is used for updating schedule relations and for updating the shdesc.miscj relation. Where it is not true, it will be as if the schedules without valid names did not exist as far as the assigner is concerned. These schedules will not be tampered with, but are unretrievable.

The assigner assumes that appropriate job description records will appear in ncjdat.descj, but is forgiving when this is not the case, allowing the user to go make needed additions in the DBU and then come back to finish his assigner run.

The assigner assumes that the DBU maintains the SUBRELUMAP field in ncjodat records, indicating the presence of data in subsidiary relations which will require deletion in the event of trimary schedule deletion. Since there are no subsidiary relations supported at this time, this feature is inoperative. Care must be made in constructing future DBU screens, however. See the asubdl routine for more information.

The assigner makes the usual use of the DBIF, scenario system, and Core data swap facilities through the usual utilities.

12.8 SUBROUTINE ABSTRACTS

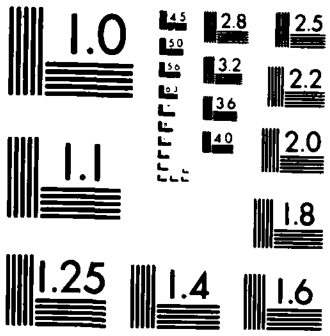
```
C     ASGN*****************************************************
$CONTROL segment=asgnd,check=3
      PROGRAM asgn
C*                                          *** ABSTRACT ***
C#PURPOSE executive for the Manual Assigner Module
C#AUDIT HISTORY
C         Densmore        15-Mar-83  AUTHOR
C         Densmore        19-Jul-83  Made Son Process of module
C#TYPE    manual assigner routine
C#COMMON BLOCKS
Cin       scrchr  screen characters
C#CALLER  Menu system through process create/activation (ASSIGN)
C#METHOD
C  executive
C#LOCAL VARIABLES
C         icmd    command index
C         isub    subcommand index
C         val     integer array giving input numeric values
C         nval    length of val array
C##
```

```
C     ASCDAY ***********************************************
$CONTROL segment=ASGNO
C$TRACE ascday;
      INTEGER*4 FUNCTION ascday(srcday,srcod,tgtcod,pfindx)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer*4 srcday
      integer srcod,tgtcod,pfindx
C*                                           *** ABSTRACT ***
C#PURPOSE   ASsigner Calculate DAY.  Use one date and
C     schedule planning factors to calculate another date.
C     Expects and outputs a clarified ddate.
C#AUDIT HISTORY
C        MSCarey        27-jun-83  AUTHOR
C#FORMAL PARAMETERS
Cin      srcday   date to use as basis
Cin      srcod    code indicating which milestone srcday is
Cin      tgtcod   code indicating which milestone is desired
Cin      pfindx   index of job description to use in /asjd/
C#COMMON BLOCKS
Cin      asoprm   outbound parameters
Cin      asjd     job descriptions
C#CALLER various assigner outbound
C#METHOD
C     Sum up the intervals between src milestone and tgt milestone
C     and add that number of days to get the output.
C##
```

```
C     ASCDSP ************************************************
$CONTROL segment=ASGNO
C$TRACE ascdsp;
      INTEGER*4 FUNCTION ascdsp(date,start,pfindx)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      integer*4 date,start
      integer pfindx
C*                                             *** ABSTRACT ***
C#PURPOSE   ASsigner Change DiSPlay date.  Alters an award date
C        so that its MM/DD are according to planning factors.
C#AUDIT HISTORY
C        MSCarey        27-jun-83  AUTHOR
C#FORMAL PARAMETERS
Cin      date     current date
Cin      start    start date for ship; new award date must be less
Cin      pfindx   location of proper planning factors in /adjd/
C#COMMON BLOCKS
Cin      asoprm   outbound parameters
Cin      asjd     job description tuple images
C#CALLER asntrb
C#METHOD
C     Move the desired mm and dd into the output variable from the
C     planning factors.  The year will be the later of the year
C     of date or the year after date; ascdsp must be within
C     12 months of date.
C##
```

AD-A150 423 ALIAS (ACQUISITION AND LOGISTICS INFORMATION AND ANALYSIS SYSTEM) MAINTEN..(U) DECISION-SCIENCE APPLICATIONS INC ARLINGTON VA M S CAREY ET AL. 6/7

UNCLASSIFIED 31 OCT 84 DSA-593-VOL-2 N00014-82-C-0813 F/G 15/5 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
C     ASCMPG ***********************************************
$CONTROL segment=ASGNO
C$TRACE asycls;
      SUBROUTINE ascmpg(mcyds,yard,nclas,lstchr,clist,cgcode)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer iyard,nclas,lstlen,cgcode(nclas)
      character*(lstchr) clist(nclas) ,yard*(mcyds)
C*                                            *** ABSTRACT ***
C#PURPOSE   Assigner CoMPlexity Group identifier.  Takes
C     a list of classes and a yard reference and identifies
C     what complexity group each class belongs to.
C#AUDIT HISTORY
C       MSCarey         19-jun-83  AUTHOR
C#FORMAL PARAMETERS
Cin      iyard    location of yard name in /asgn/
Cin      nclas    number of classes on list
Cin      lstchr   number of chars in each class name
Cin      clist    list of class names
Cout     cgcode   list of complexity-group codes for each class
C                 codes are abitrary,but same for every member
C                 of the same group
C#COMMON BLOCKS
Cin      asjd     job descriptions for each class
C#CALLER asncnv
C#METHOD
C     Make up a key for each yard-class-jobtype combination, and call
C     asgdsc to get a job description tuple into the /asjd/ buffer
C     for each.  Assign complexity-group codes based on the tuples.
C     If spreading mode is none or class, assign each class to a
C     separate complexity group and do not retrieve job desc tuples.
C##
```

```
C     ASDWRN ************************************************
$CONTROL segment=ASGNO
      SUBROUTINE asdwrn(yard,class,hull)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer hull
      character yard*8,class*10
C*                                          *** ABSTRACT ***
C#PURPOSE   ASsigner hardwire tuple Deletion WaRNing.
C#AUDIT HISTORY
C        MSCarey        01-jul-83  AUTHOR
C#FORMAL PARAMETERS
Cin      yard     name of yard for tuple about to be deleted
Cin      class    name of class of tuple about to be deleted
Cin      hull     hull number for tuple about to be deleted
C#COMMON BLOCKS
Cin      ioc      io unit numbers
C#CALLER various assigner outbound
C#METHOD
C     Write a message
C##
```

```
C     ASGNXT ************************************************
$CONTROL segment=ASGNO
C$TRACE asycls;
      SUBROUTINE asgnxt(nclas,cgcode,cptr,mclchr,numpds,cused,
     1                    hldcls,hldnum,nclash,nomore)
C*                          *** FORMAL PARAMETER DECLARATIONS ***
      integer nclas,cgcode(nclas),cptr(nclas),nclash,mclchr,numpds
      integer hldnum(nclas)
      logical cused(nclas),nomore
      character*(mclchr) hldcls(nclas)
C*                                          *** ABSTRACT ***
C$PURPOSE   ASsigner outbound Get Next complexity group.  Moves
C     bufasn lines for a related set of classes into holding buffers
C$AUDIT HISTORY
C        MSCarey        18-jun-83  AUTHOR
C$FORMAL PARAMETERS
Cin      nclas    number of classes in yard
Cin      cgcode   complexity-group idcode of each class
Cin      cptr     pointer to class record in bufasn
Cin      mclchr   max characters in class name
Cin      numpds    max periods; length of bufasn record in effect
Cio      cused    true if class(i) has already been processed
Cout     hldcls   name of class(i) with job type char
Cout     nclash   number of classes placed in hld___ buffers this call
Cout     hldnum   location on clist of class i in hld___
Cout     nomore   true if all classes in this yard have been processed
C$COMMON BLOCKS
Cio      asgn     bufasn and edit stage blocks
C$CALLER asncnv
C$METHOD
C     Look through the class list for the first unprocessed class.
C     Retrieve the record for this class and place it in hld___
C     Look through the rest of the list for classes with the same
C        complexity group code and store their bufasn records.
C$$
```

```
C      ASGPF ************************************************
$CONTROL segment=ASGNO
C$TRACE asgpf;
       INTEGER FUNCTION asgpf(sercod)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
       integer sercod
C*                                            *** ABSTRACT ***
C$PURPOSE   ASsigner outbound Get Planning Factor index.
C      Returns an index in /asjd/ of the job description
C      which most closely matches the requested job series type.
C$AUDIT HISTORY
C         MSCarey        27-jun-83  AUTHOR
C$FORMAL PARAMETERS
Cin       sercod   index indicating series type of ship (e.g. LEAD)
C$COMMON BLOCKS
Cin       asnvld   job type and series type reference
Cin       asjd     job description tuple images
C$CALLER astrb
C$METHOD
C      Convert the code to a string value.  Look for an exact match
C      in /asjd/.  If none, warn and use general purpose description.
C$$
```

```
C     ASHARD ***********************************************
$CONTROL segment=ASGNO
C$TRACE asycls;
      SUBROUTINE ashard(mxclcr,nclash,hldcls,mxcyrd,yard)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      character*(mxclcr) hldcls(nclash), yard*(mxcyrd)
      integer mxclcr,nclash,mxcyrd
C*                                            *** ABSTRACT ***
C$PURPOSE   ASsigner HARDwire tuple integration routine.
C     'Deletes' one record from /rbuf/ for each hardwire tuple.
C$AUDIT HISTORY
C        MSCarey        19-jun-83  AUTHOR
C$FORMAL PARAMETERS
Cin      mxclcr   number of characters in class name
Cin      nclash   number of classes being processed now
Cin      hldcls   names of classes being processed now
Cin      mxcyrd   max number of characters in yard name
Cin      yard     name of current yard
C$COMMON BLOCKS
Cio      asrbuf   1-ship 1-record buffer
Cout     astfr    tuple and tupfil record buffers
C$CALLER asncnv
C$METHOD
C     For each class-jobt, point to the first flagged tuple in
C     ncjodat using the cursor with the flags-only selection.
C     Return tuples for this class until there are no more.
C     For each tuple, find the record with the closest rdispd
C     in rbuf which has an exact match on the job type of the
C     tuple.  Delete this record by removing it from the pointer
C     chain.  If no record can be found with a match on job type
C     or with rfirst <= tupadj <= rlast, delete the tuple.
C     Search for subsidiary tuples and delete them also.
C$$
```

```
C     ASHTRB ***********************************************
$CONTROL segment=ASGNO
C$TRACE asycls;
      SUBROUTINE ashtrb(yard,ydchr,clchr,numpds,hldcls,
     1      hldnum,nclash,mxcls,clist)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      character*(ydchr) yard
      character*(clchr) hldcls(nclash),clist(mxcls)
      integer clchr,numpds,nclash
      integer mxcls,hldnum(nclash),ydchr
C*                                          *** ABSTRACT ***
C$PURPOSE   ASsigner To tuple Record Buffer.  Converts hld___
C     form of assignments to 1-line-per-ship representation
C     useful to the data spreader and tupfil producer.
C$AUDIT HISTORY
C        MSCarey         19-jun-83  AUTHOR
C$FORMAL PARAMETERS
Cin      ydchr    max chars in yard
Cin      yard     name of yard being processed
Cin      clchr    max chars in a class name
Cin      hldcls   name of class(i)
Cin      nclash   number of classes in hld___
Cin      numpds    dimension of hld___. Same as numper
Cin      hldnum   position of hclass on clist
Cin      mxcls    dimension of clist
Cin      clist    alphabetic list of classes in yard
C$COMMON BLOCKS
Cin      asoprm   general outbound params and variables
Cout     asrbuf   1-ship 1-record structure; output of this routine
C$CALLER asncnv
C$METHOD
C     First, run through the hld___ structure and construct asrbuf
C     and associated pointers.  Assume that the display date is to
C     be the first day of its period in each case.
C     Then process one class at a time.
C     First, load planning factors for the class.
C     If the display basis is award and time unit years, then
C     convert display dates from the first to the
C     proper date.
C     Then arrive at adjust-basis dates for each ship.
C$$
```

```
C     ASJOI ***************************************************
$CONTROL segment=ASGNO
C$TRACE asjoi;
      LOGICAL FUNCTION asjoi(string)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      character*12 string
C*                                          *** ABSTRACT ***
C$PURPOSE   Decides if the job type implied by the character
C     in position 11 of string is one of outbound interest.
C$AUDIT HISTORY
C        MSCarey        27-jun-83  AUTHOR
C$FORMAL PARAMETERS
Cin      string   class name with job type recog char attached
C$COMMON BLOCKS
Cin      asnvld   recognition character conversion
C$CALLER asycls
C$METHOD
C     Only new construction job types are of interest.
C     Job types are listed as new construction or repair
C     in a companion char variable to jtvld.  Indexing
C     first on jtvld and then on jttype gives the desired value
C$$
```

```
C      ASN1ST**************************************************
$CONTROL segment=asgni,check=3
      INTEGER*4 FUNCTION asn1st(fyear,fdate,idurat)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER fyear,fdate,idurat
C*                                         *** ABSTRACT ***
C#PURPOSE Return equivalent of FDDATE(DPFRST,IDURAT) given /asgn/ data
C#AUDIT HISTORY
C         Densmore        28-Jul-83  AUTHOR
C#TYPE    assigner date routine
C#FORMAL PARAMETERS
Cin       fyear   /asgn/ fiyear - the year of the first period
Cin       fdate   /asgn/ fidate - the number in year of first period
C                 value depends on the value of idurat (including its
C                 being undefined when idurat=1 or 2)
Cin       idurat /asgn/idurat - 1=Fyr,2=Cyr,r,3=qtr,4=month,5=week,6=day
C#COMMON BLOCKS
Cin       tddate  date data type block
C#CALLER  asnmnp, assigner outbound
C#METHOD
C  simple case statement.  An FDDATE is performed on the result
C  as insurance.
C#LOCAL VARIABLES
C         date    the result before fddate call
C         i... j... day, month numbers
C##
```

```
C     ASNADD*****asnins****************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnadd(isub,val,nval)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER isub,nval,val(nval)
C*                                         *** ABSTRACT ***
C#PURPOSE implements manual assigner add assignment command
C#AUDIT HISTORY
C         Densmore       15-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       isub    subcommand index
Cin       val     user-input values array
Cin       nval    length of val
C#COMMON BLOCKS
Cin       scrchr  screen characters
Cin/out   asgn    manual assigner blocks
C#CALLER  assign
C#METHOD
C  Checks for user and system errors.  Determines if a yard is
C  to be added, and calls newyrd if so.  Loops over period pages
C  and obtains input from user defining new assignment.  Resets
C  assignment record pointers so that the new record is inserted
C  in proper order by ship-class name.
C#LOCAL VARIABLES
C         loc     index for the new yard
C         start   first period on this page
C         len     number of periods on this page
C         last    last period on this page
C         msg     message buffer
C         class   class buffer
C         before  index of item before the one searched; 0 if first
C         item    index of item searched; 0 if not present
C         after   index of item after the one searched; 0 if last
C         valbuf  values buffer
C         codbuf  codes buffer
C         t,nil   .True.,.False. -- easier to see
C         xsec    cross section sum
C         ifree   pointer to next item in free chain (after freptr)
C                 used to set freptr at conclusion of add operation
C         look    class index for which to look; 0 if Add command
C         mval    maximum val index expected [1..2]
C         inmode  IF asnadd THEN t ELSE (nil & Assert asnins)
C         beyond  name of max val index expected [yard,class]
C##
```

```
C     ASNALO*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnalo
C*                                          *** ABSTRACT ***
C#PURPOSE allocates an assignments buffer; places it on free chain
C#AUDIT HISTORY
C         Densmore         21-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  asnadd
C#METHOD
C  Two variables store the free buffer records status: nvruse and
C  freptr.  nvruse is the record index of the first never-used direct
C  access buffer record.  freptr is a pointer (a record index) to the
C  head of a list of assignment buffer records which are free for use
C  (and probably got there by being used and then freed).
C#LOCAL VARIABLES
C         error   true if an I/O error occurred
C##
```

```
C     ASNALT*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnalt(len,valbuf,codbuf,values,codes)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER len,valbuf(len),codbuf(len),values(len),codes(len)
C*                                                *** ABSTRACT ***
C#PURPOSE Alters buffer values/codes according to buffer codes
C         so that Modification is affected.
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       len     number of periods in question
Cin/out   valbuf  new values input...undefined if corresponding
C                 code is cundef
Cin/out   codbuf  new codes input...cundef means change (valbuf,codbuf)
C                 to (0,0)...0 means change pair to (values,codes ),
C                 that is, the old values.
Cin       values  the old values
Cin       codes   the old codes
C#CALLER  asnmod
C##
```

```
C     ASNAMM***************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnamm(val,nval)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER nval, val(nval)
C*                                            *** ABSTRACT ***
C#PURPOSE Allows modification of a yard or class name
C#AUDIT HISTORY
C         Densmore        11-Jul-83  AUTHOR
C#TYPE    assigner routine
C#FORMAL PARAMETERS
Cin       val     input numeric parameters
Cin       nval    number of parameters
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  asnmod
C#METHOD
C  Determines whether yard or class name is being modified,
C  checks that line is being displayed, solicits new name,
C  checks it, and makes the change.
C#LOCAL VARIABLES
C         buffer  character variable storing new name
C         old     old value
C         mcls    true if class name being modified (else yard name)
C         msg     message buffer
C         icls    class number
C         iyrd    yard number
C         leno/b  character lengths
C         item    pointer to appropriate class
C##
```

```
C     ASNCAL*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asncal(yard)
      INTEGER yard
C#PURPOSE asks that next refresh center around mentioned yard
C#AUTHOR  Densmore 1 April 1983
C##
```

```
C     ASNCHD*************************************************
$CONTROL segment=asgno,check=3
      SUBROUTINE asnchd(bdate,nbd,tdate,prioty,ntd,ndate,nnd,mnd)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER nbd,ntd,nnd,mnd,prioty(ntd)
C     ---DDATE bdate(nbd),tdate(ntd),ndate(mnd):
      INTEGER*4 bdate(nbd),tdate(ntd),ndate(mnd)
C*                                          *** ABSTRACT ***
C$PURPOSE Accepts Buffer_DATEs, Tuples_DATEs, outputs instructions
C         for converting tdates into new dates in New_DATEs.
C$AUDIT HISTORY
C         Densmore        27-Apr-83  AUTHOR
C$TYPE    Manual Assigner routine
C$FORMAL PARAMETERS
Cin       bdate   Sorted Dates derived from the buffer assignment for a
C                 particular yard/class/period; presumably these
C                 dates are evenly spaced over the period.
Cin       nbd     number of bdates
Cin       tdate   Sorted Dates obtained from database corresponding to
C                 the old assignments in this yard/class/period.
C                 These are presumably obtained directly from the
C                 tuples in the database.
Cin       prioty  A priority for the old dates. Currently, there are
C                 only two values: 0="softwire", 1="hardwire"
Cin       ntd     length of the arrays tdate and prioty
Cout      ndate   The result array.  If ndate(i) is nonzero, then
C                 the tuple corresponding to i should be updated
C                 to contain the date ndate(i).  If ndate(i) is
C                 zero, that tuple should be removed from the
C                 database.  If nnd>ntd, then the tuples for
C                 which i>ntd should be added to the database.  If
C                 ntd>nnd, then there are exactly (ntd-nnd) ndate
C                 values which are zero.  Note result not sorted.
C                 *** It is assumed that all valid dates are > 0.
Cout      nnd     Length of ndate
Cin       mnd     Maximum allowable length for ndate
C$COMMON BLOCKS
Cin       tddate  DDATE data type block
Cin       lprnts  diagnostics
C$METHOD
C     The method is a three step process.  First, recall that the
C total number of dates we wish to be left with is nbd.  If ntd>nbd,
C then the first step is to mark for deletion the (ntd-nbd) latest
C softwire tuples.  If more must be deleted then exist, then the
C algorithm begins deleting the latest hardwire tuples.
C     The second step is to loop through and mark all the hardwire
C tuples that remain to be kept.  Let the number of such tuples be
C given by nhard.  Then at this point (nbd-nhard) dates remain to
```

```
C be specified.
C     Now, for each hardwire tuple being kept, exactly one
C softwire date in bdate must be ignored.  The one ignored at each
C step is the one "closest" to the hardwire tuple, timewise.
C (Datatype DDATE function DCLOSR is used.)  The dates remaining
C after the ones to be ignored are marked are placed in ndate at the
C appropriate spots, and processing is complete.
C#LOCAL VARIABLES
C          unmark  an unmarked state variable of type DDATE
C          delete  a marked state meaning delete this tuple
C          toogrt  a diagnostic state meaning index > maximum
C          d       delete-ndate index
C          h       hardwire-ndate set index
C          a       arbitrary ndate index
C          hard    boolean indicating now doing hardwire deletions
C          hipri   stmt function true when hi-priority is on
C          nhd     number of hardwire dates to be kept
C          ignore  flags indicating that the corresponding Buffer-
C                  DATE should not be used to set NDATE values
C          clsest  buffer-DATE index such that:
C                     BDATE(clsest) >= NDATE(h) ,
C                  but
C                     BDATE(clsest-1) < NDATE(h) ;
C                  Overflow condition indicated by clsest= 0 or nbd+1
C          low     nearest lower unignored BDATE to BDATE(clsest)
C          high    nearest higher unignored BDATE to BDATE(clsest)
C          set     either low or high value
C##
```

```
C     ASNCHK*********************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnchk(ival)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER ival
C*                                            *** ABSTRACT ***
C#PURPOSE Executes diagnostic action numbered ival
C#AUDIT HISTORY
C        Densmore       28-Mar-83  AUTHOR
C#TYPE   manual assigner utility
C#FORMAL PARAMETERS
Cin      ival    diagnostic action index
C#COMMON BLOCKS
Cin      asgn    assigner data block
C#CALLER  asnsee
C##
```

```
C     ASNCLN ************************************************
$CONTROL segment=ASGNO
      SUBROUTINE asncln
C*                      *** FORMAL PARAMETER DECLARATIONS ***
C*                                            *** ABSTRACT ***
C#PURPOSE   ASigNer CLeaN.
C           Removes any instances of historical or current jobs
C           from bufasn to ensure that no duplication in the
C           data base occurs
C#AUDIT HISTORY
C        MSCarey         16-jun-83  AUTHOR
C#FORMAL PARAMETERS
C        none
C#CALLER asnout
C#METHOD
C     Perform a selection which will cause only those tuples of
C     interest to be returned.  For each such tuple, figure out
C     which cell of bufasn it was put in and decrement that cell
C     Check the cell value to see if it is now < 0, and warn
C     the user that current/historical assignments may not be
C     changed with the assigner.
C#LOCAL VARIABLES
C        cursor   (1) for ncjodat.histj, (2) for .currj  selections
C##
```

```
C      ASNCLR***********************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnclr
C*                                                  *** ABSTRACT ***
C#PURPOSE clears screen for assigner module
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  Several ASN routines
C##
```

```
C     ASNCMD*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asncmd(icmd,isub,val,nval,mval)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER icmd,isub,nval,mval,val(mval)
C*                                           *** ABSTRACT ***
C#PURPOSE Read user command and decode it
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cout      icmd    major command index (tied to /scrchr/)
Cout      isub    subcommand index (/scrchr/)
Cout      val     values array -- set of integers separated
C                  on input by periods here.
Cout      nval    length of val
Cin       mval    maximum length of val
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
Cin       scrchr  screen characters
C#CALLER  assign
C#METHOD
C  <Command-string>  ::=  <Command> <Subcommand> [<Num> {<Delim> <Num>}]
C  <Command>         ::=  (one of the /scrchr/ characters)
C  <Subcommand>      ::=  <Command>  |  <Null>
C  <Num>             ::=  (an integer)
C
C                     200
CBLANK       BLANK     |    BLANK        DIGIT  ,->DOT or COMMA>-, BLANK
C | ^         | ^  .->EOL<-  | ^           | ^  /       BLANK    C | |         |
| /          | |
C v |         v |/            v |
C 100-->CMD-->110-->SubCMD-->120-->DIGIT-->130-->BLANK-->140-->DOT-->150
C              |                            ^ ^C              |
| | C              '---->DIGIT--------------->-' |  C
|                              |
C Out of every state is an implicit "ANY     '-<--------DIGIT<--------'
C OTHER CHARACTER" whose vector leads to
C an error state, which returns to $10.
C
C#LOCAL VARIABLES
C         csave   command indices corresponding to commands
C                 for which saves (CALL cmnsav) must be done
C##
```

```
C     ASNCNV ************************************************
$CONTROL segment=ASGNO
      SUBROUTINE asncnv(numpds)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      integer numpds
C*                                          *** ABSTRACT ***
C*PURPOSE   ASsigNer CoNVert.  Converts bufasn into tupfil.
C     Basically, input is as displayed by assigner, output
C     is tuples as found in the data base.
C*AUDIT HISTORY
C        MSCarey         16-jun-83  AUTHOR
C*FORMAL PARAMETERS
Cin        numpds  number of periods; same as numper
C*CALLER asnout
C*METHOD
C  *  Loop over yards in bufasn.
C (1) For each yard, make a list of classes and bufasn record
C     numbers for those which have non-zero assignments. Sort the list.
C (2) Search the list for classes belonging to the same spreading-group
C     [menu option complexity-group].  Load all classes belonging to
C     the same group into a holding buffer, and mark them as processed
C     on the list.  Unless the user has chosen the complexity-group
C     spreading option, this function will return one class at a time
C     until a yard is exhausted.  Load schedule planning factors
C     for each class of interest into a companion buffer.
C (3) For each group, loop over columns in the holding buffer.
C     For each column,
C     loop over rows, decrementing each cell by one and constructing
C     a record for the tuple buffer.  Attention is paid to job series
C     type special characters here.  The dates put into the tuple
C     buffer should be adjustment-basis dates, which are arrived at
C     from the display-basis dates using the planning factors.  These
C     dates are therefore the earliest dates allowable after the
C     date-spreading process if display-date integrity
C     is to be maintained.
C     NOTE EXCEPTION: if display basis is awards, and time units are
C     years, then take the day of award in a given year from the
C     job desc record, rather than assuming it to be the first day
C     of the year.  Then calculate the adjdat's as usual.
C (4) If the user has specified date-spreading in his calling
C     parameters, conduct the date spreading here.
C     First spread the dates in the tuple buffer completely evenly,
C     also assigning a first allowable and last allowable date for each
C     Then go through EACH CLASS sequentially; on finding a date
C     earlier than its
C     limit, add the amount needed to bring it up to the limit to
C     every item in the class.  On finding a date later than its
C     limit, look backwards and forwards for all dates with the same
```

```
C     limits (i.e. from the same display period) and spread
C     within that period evenly.  Now recombine the classes and check
C     for instances of identical adjdat's.  If found, calculate the
C     mean interval for the nearest # ships and add/subtract half this
C     amount to each IF this will not violate the period limits.
C     Decide which to add or subtract to depending on which is closer
C     to upper/lower period limits.
C (5) Now integrate in any hardwire tuples in the data base.
C     SELECT @ BY yard,class WHERE scenario=_cursen_ and flag="up"
C     For each class now in tupbuf, calc to first matching tuple;
C     if any, then find tupbuf records with closest adjdat, and mark
C     them gone;
C (6) Now get the rest of the dates for softwires, based on the spread
C     adjdat's.  Also construct the rest of each tuple and put it into
C     tupfil.  Use standard planning factors except for award:
C     if adjbasis is award, use as calculated;
C     IF tunit is years then calc from factors and set to next-earliest
C        desc-date
C     ELSE calc from factors and set to first of period it's in, based
C        on time_units
C (7) On end of busasn for this yard, update data base.
C#LOCAL VARIABLES
C        clist    list of classes in current yard
C        cgcode   complexity-group each class is in
C        cptr     pointer to bufasn record for each class
C        nclas    number of classes in current yard
C        cused    true if a class has been processed
C        hldcls   class name of each class in hld___
C        hldval   per-period assignments for each class in a comp-grp
C        hldcod   per-period codes for each class in a comp-grp
C        nclash   number of classes in the hld___ buffers now
C##
```

```
C     ASNCOD************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asncod(newcod,oldcod)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
        INTEGER oldcod,newcod
C*                                            *** ABSTRACT ***
C#PURPOSE conditionally sets old job series code id for asndbi
C#AUDIT HISTORY
C        Densmore        24-Jun-83  AUTHOR
C#FORMAL PARAMETERS
Cin  newcod  tcode from the latest tuple
Cio  oldcod  the relevant code from this scen/yd/cld/period
C##
```

```
C     ASNCPY*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asncpy(val,nval,move,succes)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER nval, val(nval)
      LOGICAL succes,move
C*                                               *** ABSTRACT ***
C#PURPOSE Makes new copies of yards or classes
C#AUDIT HISTORY
C         Densmore        08-Jul-83  AUTHOR
C#TYPE    assigner routine
C#FORMAL PARAMETERS
Cin       val     input values
Cin       nval    number of input values
Cin       move    True if the purpose of the copy call is to
C                 perform a move
Cout      succes  True if command executed successfully
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  asnrlc
C#METHOD
C  Split into two parts: a copy-yard part and a copy-class part.
C       The copy-yard part is implemented in yrdcpy.
C       The copy-class part does routine checks, loops down to the
C  fromclass data, and then holds it.  Then asnfnd is called; the
C  remainder is similar to the code in asnadd -- hard to make into
C  a subroutine because variables must be held in limbo around
C  sections of code which are different for the two applications.
C#LOCAL VARIABLES
C         buffer  holds old assignment
C         i       do index
C         from    old or from yard location
C         loc     new or to yard location
C         ifree   pointer to free chain after class copy
C         class   holds old assignment class name
C         msg     holds messages to be sent to user via asnpro-mpt
C         firstm  n/o/ /l/o/n/g/e/r/ /u/s/e/d
C##
```

```
C      ASNDBI*******************************************************
$CONTROL segment=asgni,check=3
      SUBROUTINE asndbi
C*                                                    *** ABSTRACT ***
C#PURPOSE Recovers ASN data from relations
C#AUDIT HISTORY
C          Densmore        07-Apr-83  AUTHOR
C          Densmore        06-May-83  To begin looping thru DB
C#TYPE    manual assigner utility
C#COMMON BLOCKS
Cin       asgn    assigner data block
C#CALLER  asnini
C#METHOD
C##
```

```
C     ASNDBR **************************************************
$CONTROL segment=ASGNO
C$TRACE asndbr;
      SUBROUTINE asndbr(numyds,mcyds,yard,nclas,tupfst,lstcal,
     1                  clist,mccls)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer numyds,mccls,mcyds,nclas,tupfst(nclas)
      character*(mcyds) yard
      character*(mccls) clist(nclas)
      logical lstcal
C*                                              *** ABSTRACT ***
C#PURPOSE   ASsigNer Data Base tuple Replacement routine.
C     Updates the tuples in the data base for the given yard
C     using the assignments implied by the current state of
C     bufasn.
C#AUDIT HISTORY
C        MSCarey        03-jul-83  AUTHOR
C#FORMAL PARAMETERS
Cin      numyds   number of yards in bufasn; if zero, clean bufasn
Cin      mcyds    length of yard
Cin      yard     yard to be updated
Cin      nclas    number of class-jobtypes in tupfil
Cin      tupfst   pointer to first tupfil record for each clas-jobt
Cin      lstcal   true if this is the last call to asndbr; in this
C                 case, processing should go on until the end of tuples
C                 for this scenario so that any trailing deleted
C                 assignments are caught.
Cin      clist    list of classes found in bufasn for this yard
Cin      mccls    length of clist class names.
C#COMMON BLOCKS
Cin      asoprm   outbound paramters
Cin      asnocr   outbound cursors
Cio      astfr    buffer for relation and tupfil records
C#CALLER asncnv
C#METHOD
C     Basically, series of cases.  There is always a current tuple
C     and a current tupfil record.  The actions which may be taken
C     are to update the tuple using the record, to add the record
C     to the relation, to skip to the next tuple, and to delete the
C     tuple.  Which is appropriate depends on a comparison of the
C     values of the yard,class, and jobtype fields in the tuple and
C     tupfil holding buffers.  Both tuples and tupfil records are
C     assumed to arrive in their holding areas sorted by yard, class,
C     and jobtype.
C#LOCAL VARIABLES
C        eofil    true if no more records in tupfil
C        lstyrd   yard name of tupfil record prev to current record
C        lstcls   class "
```

```
C         lstjob   job typ "
C         next     record number in tupfil of next record
C         iclas    class-job type on clist now being processed
C##
```

```
C     ASNDEL*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asndel(isub,val,nval)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER isub,nval,val(nval)
C*                                              *** ABSTRACT ***
C#PURPOSE implements manual assigner delete assignment command
C#AUDIT HISTORY
C         Densmore       15-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       isub    subcommand index
Cin       val     user-input values array
Cin       nval    length of val
C#COMMON BLOCKS
Cin       scrchr  screen characters
Cin/out   asgn    manual assigner blocks
C#CALLER  assign
C#METHOD
C  If an assignment index is given, deletes that assignment.  Checks
C  first to see if it is the last.  If so or if no assignment given,
C  the yard is deleted after prompting to make sure.
C#LOCAL VARIABLES
C         msg     message buffer for asnpro/asnec
C         loc     yard index
C         before  pointer to assignment record before delete item
C         item    pointer to delete item
C         after   pointer to item's successor
C         next    do index
C##
```

```
C     ASNDOT***********************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asndot(getcls,alwdel,len,class,values,codes)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      LOGICAL getcls,alwdel
      INTEGER len,values(len),codes(len)
C *** CHARACTER*12 class -- 12==mccls (given in /asgn/)
C*                                             *** ABSTRACT ***
C#PURPOSE Types the dots for prompting of assignments input,
C         and accepts and verifies the input.  Output to values
C         and codes array in decoded form.
C#AUDIT HISTORY
C         Densmore       17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       getcls  .T. if shipclass should be prompted for and
C                 received as input
Cin       alwdel  .T. if DELetion of assignments should be allowed
C                 for modifications purposes...input form is "--"
Cin       len     number of periods over which input is expected
Cout      class   output shipclass, char*mccls...not output unless
C                 the getcls flag is true.
Cout      values  output values
Cout      codes   output codes
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  asnadd,asnmod
C#METHOD
C  Prompts for add and modify commands, via dots which delineate
C  proper placement of each number/code sequence (and also the
C  class name, if requested).  for each period, the following
C  sequence of characters (3 for each period) are valid; the
C  first character must always be a blank  (CU means cundef):
C     Char-1   Char-2   Char-3   Value   Code
C     ------   ------   ------   -----   ----
C      blank    blank    blank     0       0
C        "        "      zero      0     CU IF alwdel ELSE 0
C        "        "      Digit3    D3    cdidef
C        "      zero     blank     0     CU IF alwdel ELSE 0
C        "        "      zero      0     CU IF alwdel ELSE 0
C        "      Code2    Digit3    D3    C2
C        "      Digit2   blank     D2    cdidef
C        "        "      Digit3  10*D2+D3 cdidef
C#LOCAL VARIABLES
C         m/c/c/l/s/l  -- variable now in common
C         cdots   prompt for class name (either blank, or dots)
C         dot2    prompt for each period assignment input (" ..")
C         number  the string of digits 0 through 9
```

```
C         buffer  location where user's input is accepted
C         b1,b2,b3 each character of a given assignment entry
C         iper    the current period index (1..len)
C         nchar   the number of blanks before first prompt dot
C##
```

```
C     ASNDWN***********************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asndwn(toploc,topind)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER toploc,topind
C*                                            *** ABSTRACT ***
C#PURPOSE performs "DOWN" (following page) command
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       toploc  value desired for topyd
Cin       topind  value desired for topidx
C#COMMON BLOCKS
Cin/out   asgn    assigner data
C#CALLER  asnfol,asnrfh
C#METHOD
C  If BOTTOM, sets low limits to last assignment index, then
C  computes upper limits.  If DOWN, sets upper limits to former
C  lower limits and recomputes lower limits; if it crosses the
C  bottom then BOTTOM command is performed.
C
C  Also, this routine always recomputes the page number npagev.
C##
```

```
C     ASNEC***************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnec(what,value,type,text)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER what,value,type
      CHARACTER*255 text
C*                                           *** ABSTRACT ***
C#PURPOSE Assigner Error in Command reporter
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       what    what was in error; refers to explicit errors
C                 in the input user command, rather than to logic
C                 errors caused by faulty command order or intent.
C                    0=none
C                    1=major command index
C                    2=subcommand index
C                   -N=val array element !N!
Cin       value   Value code for the item in error; the value of the
C                 index or the value of the val array element.
C                 Undefined if what=0.
Cin       type    Error type: 0=some external system error or error
C                 about which no further information should be printed.
C                 1=explicit user-input command error.  2=user stressed
C                 assigner up against some limitation (array bound,etc.)
Cin       text    Delimited text string which should be printed to
C                 assist in describing the cause of the problem.
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  various ASN routines
C##
```

```
C     ASNEND***************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnend
C*                                                *** ABSTRACT ***
C#PURPOSE Prepares ASGN for STOP (End of process)
C#AUDIT HISTORY
C         Densmore        19-Aug-83  AUTHOR
C#TYPE    assigner
C#COMMON BLOCKS
Cin       asgn
C#CALLER  asgn
C#METHOD
C  Currently just closes the files which are open.
C##
```

```
C     ASNEOI*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asneoi
C*                                              *** ABSTRACT ***
C#PURPOSE reverts I/O switches upon EOF on 'inasn' in /asgn/
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
Cin/out   ioc     input/output variables
C#CALLER  asncmd,asndot,newyrd,remyrd
C##
```

```
C     ASNFND*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnfnd(loc,class,look,before,item,after)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER loc,look,before,item,after
C *** CHARACTER*12 class  -- 12==mccls /asgn/
C*                                          *** ABSTRACT ***
C#PURPOSE Locates a particular class within a given yard
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       loc     yard index
Cin       class   name of class  char*12
Cin       look    if location of new cls  is by index, look is nonzero
C                 and gives that index: means "ins before class LOOK"
Cout      before  pointer to assignment record preceding item
C                 0 if item is first
Cout      item    pointer to assignment record with given class
C                 0 if item not found -- before/after give
C                 proper location for sorted order
C                 If item <> 0 then caller may assume that a
C                 GETASN has been done on item.
Cout      after   pointer to assignment record succeeding item
C                 0 if item is last
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  asnadd
C##
```

```
C     ASNFOL*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnfol(isub,val,nval)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER isub,nval,val(nval)
C*                                          *** ABSTRACT ***
C#PURPOSE implements manual assigner follow (down) page command
C#AUDIT HISTORY
C         Densmore       15-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       isub    subcommand index
Cin       val     user-input values array
Cin       nval    length of val
C#COMMON BLOCKS
Cin       scrchr  screen characters
Cin/out   asgn    manual assigner blocks
C#CALLER  assign
C##
```

```
C     ASNGOQ ***********************************************
$CONTROL segment=ASGNO
C$TRACE asngoq;
      LOGICAL FUNCTION asngoq(idum)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
      integer idum
C*                                                *** ABSTRACT ***
C#PURPOSE   ASsigNer Get Outbound Quit response.  Prompts
C     user before assigner outbound processing commences to
C     see if results of session should just be thrown away.
C#AUDIT HISTORY
C        MSCarey        27-jun-83  AUTHOR
C#FORMAL PARAMETERS
Cin      idum     dummy required by FORTRAN
C#COMMON BLOCKS
Cin      ioc      io unit assignments
C#CALLER asnout
C#METHOD
C     Print an explanatory message and call yesno
C#LOCAL VARIABLES
C        none
C##
```

```
C     ASNHLP*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnhlp(isub,val,nval)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER isub,nval,val(nval)
C*                                          *** ABSTRACT ***
C#PURPOSE retrieves help text for ASN
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       isub    subcommand index
Cin       val     values array
Cin       nval    length of val
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  assign
C##
```

```
C     ASNHUL ***********************************************
$CONTROL segment=ASGNO
      SUBROUTINE asnhul
C*                     *** FORMAL PARAMETER DECLARATIONS ***
C*                                           *** ABSTRACT ***
C#PURPOSE   ASsigNer HULl number reset.  Makes sure each ship
C        in the current scenario has an unique and reasonable
C        hull number
C#AUDIT HISTORY
C        MSCarey         20-jun-83  AUTHOR
C#FORMAL PARAMETERS
C        none
C#COMMON BLOCKS
C        none
C#CALLER asnout
C#METHOD
C     Get the maximum hull number for each class in ncjoprj
C     from the historical/current relations.  Execute command
C     file NEWHUL.RPROCS to do this, first writing the current
C     scenario key into it for each of the three schedule files.
C#LOCAL VARIABLES
C        none
C##
```

```
C     ASNINI***************************************************
$CONTROL segment=asgni,check=3
      SUBROUTINE asnini(abort)
      LOGICAL abort
C*                                         *** ABSTRACT ***
C#PURPOSE Manual Assigner Initialization Routine
C#AUDIT HISTORY
C          Densmore        07-Apr-83  AUTHOR
C#TYPE    Manual assigner routine
C#FORMAL PARAMETERS
Cout  abort  true if user selects abort
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  assign
C#METHOD
C  Initialization is divided into three steps.  The first is
C  to set up hardwire values.  Next, certain variables are input
C  from a file (FILE04).  Lastly, the DESC and JOB relations are
C  consulted to obtain assigner data block initial values.
C##
```

```
C     ASNLBL*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnlbl(ydlbl,clslbl,total,start)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      LOGICAL ydlbl,clslbl,total
      INTEGER start
C*                                           *** ABSTRACT ***
C#PURPOSE print top two rows of assigner display -- period rows
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       ydlbl   .T. if "yard" should be printed
Cin       clslbl  .T. if "shipclass" should be printed
Cin       total   .T. if total for the rows will be printed
Cin       start   index of first period to be printed
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  asnadd,asnmod,asnrfh
C##
```

```
C     ASNLEV ***********************************************
$CONTROL segment=ASGNO
      SUBROUTINE asnlev(quit)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      logical quit
C*                                             *** ABSTRACT ***
C#PURPOSE   ASsigNer LEaVe.  Cleans up assigner files and
C        relations before return to the menu system.
C#AUDIT HISTORY
C        MSCarey          22-jun-83  AUTHOR
C#COMMON BLOCKS
Cio      asnocr   cursors for the assigner
Cin      asoprm   assigner outbound parameters
C#CALLER asnout
C#METHOD
C     Many calls to rvclos and filcls
C##
```

```
C     ASNLFT***************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnlft(isub,val,nval)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER isub,nval,val(nval)
C*                                          *** ABSTRACT ***
C#PURPOSE implements manual assigner left page command
C#AUDIT HISTORY
C         Densmore       15-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       isub    subcommand index
Cin       val     user-input values array
Cin       nval    length of val
C#COMMON BLOCKS
Cin       scrchr  screen characters
Cin/out   asgn    manual assigner blocks
C#CALLER  assign
C#METHOD
C  Determines if edge command is needed; pages left/right
C  accordingly.
C##
```

```
C     ASNLPR*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnlpr(isub,val,nval)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER isub,nval,val(nval)
C*                                          *** ABSTRACT ***
C#PURPOSE PRINT command executive
C#AUDIT HISTORY
C         Densmore        26-May-83  AUTHOR
C#TYPE    Manual assigner routine
C#FORMAL PARAMETERS
Cin       isub    subcommand index
Cin       val     user-input values array
Cin       nval    length of val
C#COMMON BLOCKS
Cin/out   asgn    manual assigner blocks
C#CALLER  assign
C#METHOD
C  Determines validity of input.  Given valid values, it then
C  prints to a (reset) outasn via asnrfh, manipulating the
C  top/low yards/indexes as required to get whole yards on
C  a page.
C#LOCAL VARIABLES
C         start   first yard to print
C         stop    last yard to print
C         ydfrst  first yard to be printed on this page
C         ixfrst  first index to be printed on this page
C##
```

```
C     ASNMOD*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnmod(isub,val,nval)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER isub,nval,val(nval)
C*                                          *** ABSTRACT ***
C#PURPOSE implements manual assigner modify assignment command
C#AUDIT HISTORY
C         Densmore       15-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       isub    subcommand index
Cin       val     user-input values array
Cin       nval    length of val
C#COMMON BLOCKS
Cin       scrchr  screen characters
Cin/out   asgn    manual assigner blocks
C#CALLER  assign
C#METHOD
C  Locates yard/assignment and loops over periods
C#LOCAL VARIABLES
C         msg     message buffer for prompts
C         valbuf  values buffer
C         codbuf  codes buffer
C         diff    difference between buffer and old valasn page
C         loc     yard location index
C         idx     assignment index
C         change  SUM(diff[i])
C         start   starting period on current page
C         len     number of periods on current page
C         last    last period on current page
C         item    pointer to assignments buffer beine modified
C         t,nil   easier to read than TRUE/FALSE
C         begin   T on first iteration of #60, F otherwise
C##
```

```
C     ASNMOV*****************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnmov(val,nval)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER nval, val(nval)
C*                                              *** ABSTRACT ***
C#PURPOSE Implements Relocate Move command
C#AUDIT HISTORY
C         Densmore        08-Jul-83  AUTHOR
C#TYPE    assigner routine
C#FORMAL PARAMETERS
Cin       val     numeric values
Cin       nval    number of values
C#COMMON BLOCKS
Cin       asgn    assigner data block
C#CALLER  asnrlc
C#METHOD
C  See inline comments
C  Implemented via copy followed by delete
C##
```

```
C     ASNOUT ************************************************
$CONTROL segment=ASGNO
      SUBROUTINE asnout
C*                        *** FORMAL PARAMETER DECLARATIONS ***
C*                                              *** ABSTRACT ***
C#PURPOSE   ASsigNer OUTbound.  Converts edit-able assigner
C           data base to RELATE data base format.
C#AUDIT HISTORY
C       MSCarey         16-jun-83  AUTHOR
C#FORMAL PARAMETERS
C       none
C#COMMON BLOCKS
C       asoprm   general outbound
C#CALLER assign
C#METHOD
C     (1) Prompt the user to see if he wants to exit without saving
C         the results of his assigner session.  Intended to allow
C         users to peruse assignments without having to pay the
C         execution-time price of the outbound leg if they have made
C         no changes.  Remind the user at this point that no changes
C         to repair or historical/current assignments will be saved.
C     (2) Remove from bufasn any instances of historical/current
C         assignments, where an instance is defined as an increment
C         in any bufasn cell.
C     (3) Convert bufasn into a set of tuples, one per ship, which
C         is merged with any flagged ('hardwire') tuples from the
C         RELATE data base.  Take care of uniform spreading of
C         construction schedule dates here.
C     --- Make an update pass on the data base so that the set of
C         tuples in the data base is identical to that generated
C         in step 3.  Note that subsidiary tuples must be purged
C         for purged hardwire tuples.
C     (5) Make another update pass at the data base to arrive at
C         a set of consistent hull numbers, unique for each ship.
C         'Hardwire' tuples retain their own hull numbers.
C     (6) Clean up and exit.
C#LOCAL VARIABLES
C       quit     true if user wants to throw away bufasn
C##
```

```
C     ASNPOP*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnpop(isub,val,nval)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER isub,nval,val(nval)
C*                                          *** ABSTRACT ***
C#PURPOSE performs termination code when exit requested from ASN
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE     manual assigner routine
C#FORMAL PARAMETERS
Cin       isub    subcommand index
Cin       val     input values array
Cin       nval    length of val
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  assign
C##
```

```
C     ASNPRN**************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnprn(typcls,total,start,idx)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER start,idx
      LOGICAL typcls,total
C*                                          *** ABSTRACT ***
C#PURPOSE Prints a line from the buffer during display formatting
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       typcls  .T. if shipclass name should be printed
Cin       total   .T. if total for row should be printed
Cin       start   index of first period on the row
Cin       idx     assignment index (for printing purposes only)
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  asnmod,asnrfh
C##
```

```
C     ASNPRO************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnpro(text)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER*255 text
C*                                          *** ABSTRACT ***
C#PURPOSE prints prompt text conditionally
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       text    prompt text
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  various ASN routines
C##
```

```
C     ASNPRV*****************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnprv(isub,val,nval)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER isub,nval,val(nval)
C*                                                *** ABSTRACT ***
C#PURPOSE implements manual assigner previous (up) page command
C#AUDIT HISTORY
C         Densmore       15-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       isub    subcommand index
Cin       val     user-input values array
Cin       nval    length of val
C#COMMON BLOCKS
Cin       scrchr  screen characters
Cin/out   asgn    manual assigner blocks
C#CALLER  assign
C#METHOD
C  If TOP then set top limits and locate lower limits.
C  If UP then set lower limits from former upper limits and locate
C  upper limits; if top limits are crossed, then a TOP command
C  is done.
C##
```

```
C     ASNRDC*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnrdc(msg,hlptxt,case,proc,buffer,len,vld)
C*                   *** FORMAL PARAMETER DECLARATIONS ***
      PARAMETER lcase=4, lproc=4
C
      INTEGER len
      LOGICAL vld
      CHARACTER msg*255,hlptxt*8,case*lcase,proc*lproc,buffer*(len)
C*                                            *** ABSTRACT ***
C#PURPOSE Semi-general purpose input routine for assigner
C         Routine name means ASsigNer ReaD Character string
C#AUDIT HISTORY
C         Densmore        14-Jul-83  AUTHOR
C#TYPE    assigner routine
C#FORMAL PARAMETERS
Cin       msg     DTS message to print as a prompt before reading
Cin       hlptxt  Char*8 help text -- not needed if iproc=3
Cin       case    character case flag; first 4 characters significant
C                  'CLASS    ' :  set case for class name
C                  'YARD     ' :  set case for yard name
C                  'UPPER    ' :  set case to uppercase
C                  'LOWER    ' :  set case to lowercase
C                  'NOCASE   ' :  do not change case of input
Cin       proc    Process flag; first 4 characters significant
C                  'READ     ' :  Read buffer; check & process it
C                  'NULLREAD' :  Like READ, + allow null input
C                  'CHECK    ' :  Check buffer, process it
C                  'PROC     ' :  Only process buffer
Cin/out   buffer  "in" only if proc isn't READ; this is the buffer
C                  which is optionally input and checked
Cin       len     number of characters in buffer
Cout      vld     False if buffer not valid (like for EOF) or for POP
C#COMMON BLOCKS
Cin       asgn    assigner data block
Cin       scrchr  screen characters
C#CALLER  various
C#METHOD
C#LOCAL VARIABLES
C         cases   character-case options
C         procs   procedural options
C         alwrd   stmt fn means case is READ or NULLREAD
C##
```

```
C     ASNREF*****************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnref
C#PURPOSE asks for the regular screen refresh
C#AUTHOR  Densmore 1 April 1983
C##
```

```
C     ASNREO*****************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnreo(ormode)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER ormode
C*                                              *** ABSTRACT ***
C#PURPOSE Conditionally reorders classes in each yard with
C         respect to their input order.
C#AUDIT HISTORY
C         Densmore        23-Jun-83  AUTHOR
C#TYPE    assigner inbound routine
C#FORMAL PARAMETERS
Cin       ormode  Zero bit means alphabetic ordering desired;
C                 One bit means output tuples in order of input;
C                    Bit 1: for sort order of shipclasses...
C                 shipclass order is ignored by asntpx, so order is
C                 always input alphabetic.  This routine determines
C                 and accomplishes class reordering if necessary
C                    Bit 2: for sort order of shipyards... shipyard
C                 order is currently ignored.
C                 If insufficient space exists to order a yard's
C                 classes alphabetically, the order mode is changed
C                 to input-order-for-classes by force.
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
Cin       const   for llarge
C#CALLER  asndbi
C#METHOD
C  Recover ordering values from class buffer lines and save the pointers
C  to each line.  Use jhash to find new ordering.  Set nextp values.
C#LOCAL VARIABLES
C         omin,omax  smallest/largest orders value
C         ptrs       the pointers for each yard
C         orders     the order numbers for each yard; from shpord;
C                    used as data in the jhash call
C         mbuf       one more than the maximum number of classes
C                    possible in a yard -- the last one is used to
C                    set the end class buffer line's nextp to zero
C         item       pointer to current class buffer line
C         loc        index to current yard
C         count      number of class buffer lines in this yard
C         jh         ih(i)
C         jh1        ih(i+1)
C         i          do loop index from 1 to count
C         ih         output from jhash giving reordering
C         alpha      true if class sort mode is alphabetic
C         toobig     true if insufficient space to sort a yd's clses
C##
```

```
C     ASNRFH**************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnrfh(isub,val,nval)
C*                 *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER isub,nval,val(nval)
C*                                        *** ABSTRACT ***
C#PURPOSE ReFresH display on screen for ASN assigner module
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       isub    subcommand index
Cin       val     values array
Cin       nval    length of val
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  various ASN routines
C##
```

```
C     ASNRLC***************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnrlc(isub,val,nval)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER isub,nval, val(nval)
C*                                              *** ABSTRACT ***
C#PURPOSE Implements Relocate command
C#AUDIT HISTORY
C         Densmore        08-Jul-83  AUTHOR
C#TYPE    assigner high level routine
C#FORMAL PARAMETERS
Cin       isub    subcommand code
Cin       val     numeric values
Cin       nval    number of values
C#COMMON BLOCKS
Cin       scrchr  screen characters
C#CALLER  assign
C##
```

```
C     ASNSEE**************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnsee(val,nval)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER nval,val(nval)
C*                                         *** ABSTRACT ***
C#PURPOSE interpret the '??' command -- diagnostic switching
C#AUDIT HISTORY
C         Densmore        28-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       val(nval)  values input from user; interpreted as menu subcmds
C#COMMON BLOCKS
Cin       asgn    assigner data block
C#CALLER  asnhlp
C##
```

```
C     ASNSWP***********************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnswp(val,nval)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER nval, val(nval)
C*                                          *** ABSTRACT ***
C#PURPOSE Implements Relocate Swap command
C#AUDIT HISTORY
C         Densmore        08-Jul-83  AUTHOR
C#TYPE    assigner routine
C#FORMAL PARAMETERS
Cin       val     numeric values
Cin       nval    number of values
C#COMMON BLOCKS
Cin       asgn    assigner data block
C#CALLER  asnrlc
C##
```

```
C     ASNTUP ************************************************
$CONTROL segment=ASGNO
      SUBROUTINE asntup(cursor,eof)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer cursor
      logical eof
C*                                          *** ABSTRACT ***
C#PURPOSE   Reads the next tuple on cursor, which must always
C     be open on ncjodat.projj
C#AUDIT HISTORY
C        MSCarey       01-jul-83  AUTHOR
C#FORMAL PARAMETERS
Cin      cursor   relate cursor number
Cout     eof      true if end of relation found, or if scenario
C                 in found tuple not same as current scenario
C#COMMON BLOCKS
Cin      asoprm   outbound parameters
Cio      astfr    tuple/record holding buffers
Cin      fld05    field list for ncjodat.projj
Cin      scenar   scenario info
C#CALLER asndbr
C#METHOD
C     Read the tuple and check if scenario is different from the
C     current one; if so then eof.
C##
```

```
C     ASNUNL*************************************************
$CONTROL segment=asgni,check=3
      SUBROUTINE asnunl(tuple,yard,class,series,scnaro,date,order,type)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      CHARACTER scnaro*12
C *** CHARACTER yard*mcyds, class*mccls, series*mccd, type*mcjt
      INTEGER tuple(1)
      INTEGER*4 date,order
C*                                              *** ABSTRACT ***
C#PURPOSE Unloads the tuple, in char form, into constituent parts
C#AUDIT HISTORY
C         Densmore        22-Jun-83  AUTHOR
C#TYPE    assigner inbound routine
C#FORMAL PARAMETERS
Cin       tuple   integer version of the tuple
Cout      yard,class,scnaro    yard/class/scenario names
Cout      series  job series code name, like Lead, etc.
Cout      type    job type, like NewCon, etc.
Cout      date    RELATE date on which the job is marked (one of
C                  award,start,keel,launch,delivery dates)
Cout      order   the ASN-Order value which determines input order
C#COMMON BLOCKS
Cin       asgn    assigner data block, used for parameter values
Cin       asnvld  assigner valid lists, used for parameter values
C#CALLER  asntpx,nxtcls
C#METHOD
C  Unloads items into the output variables in the order they are
C  found in the tuple to begin with.  Note the type conversions
C  that are often necessary.  The parameters tup... indicate the
C  lengths alloted the various values within the tuple itself.
C  The lengths of the output variables are determined from common
C  block parameter statements.  See asntpi for input tuple order.
C#LOCAL VARIABLES
C         tup...      *2 lengths of various tuple element values
C         cbuf,ibuf   buffer to avoid assignment type conversion
C         ifour,jfour buffer to avoid conversion
C##
```

```
C     ASNVAL**************************************************
$CONTROL segment=asgni,check=3
      SUBROUTINE asnval
C*                                          *** ABSTRACT ***
C#PURPOSE Removes invalid yards/classes under current scenario
C         so that an old buffer is consistent with new constraints
C#AUDIT HISTORY
C         Densmore        03-June-83 AUTHOR
C#TYPE    assigner routine
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  asnini
C#METHOD
C  First, loop over yards and call valyrd for each to check that
C  its appearance in the buffer is allowed.  If so, then go on
C  to the next yard until EOY.  If not, then remyrd effectively
C  increments the yard index.
C
C  Next, perform a similar loop for classes in each yard.  One
C  caveat is the possibility that the last class in a yard might
C  be deleted.
C#LOCAL VARIABLES
C         iyard   yard index
C         old     old yard or class index as displayed on last refrsh
C         leny    actual character count of yard name
C         valid   true if there have been no yard/class deletions yet
C                 (ie. all present are valid...used for header print)
C         before  pointer to buffer record before current record
C         item    pointer to current record
C         after   pointer to next record
C##
```

```
C     ASNWID*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnwid(start,len,last)
C*                   *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER start,len,last
C*                                        *** ABSTRACT ***
C#PURPOSE accepts start period index, computes len and last
C#AUDIT HISTORY
C          Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       start    first period index
Cout      len      last-start+1
Cout      last     last period index -- MIN(numper,start+mperh-1)
C#COMMON BLOCKS
Cin/out   asgn     assigner data block
C#CALLER  various ASN routines
C##
```

```
C     ASNYRD************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE asnyrd(loc,name,len)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER loc,len
C *** CHARACTER name*12    12==mcyds -- /asgn/
C*                                         *** ABSTRACT ***
C#PURPOSE prints yard name/index and the --+--+--+ stuff
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       loc     yard index
Cin       name    yard name
Cin       len     number of periods
C#CALLER  asnrfh
C##
```

```
C     ASOCMP ***********************************************
$CONTROL segment=ASGNO
C$TRACE asocmp;
      INTEGER FUNCTION asocmp(rvalue,fvalue,len)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      integer len
      character*(len) rvalue,fvalue
C*                                           *** ABSTRACT ***
C#PURPOSE   ASsigner Outbound field CoMParison utility.  Returns
C     -1 if rvalue > fvalue, 0 if they are equal, 1 if rvalue<fvalue
C#AUDIT HISTORY
C        MSCarey        03-jul-83  AUTHOR
C#FORMAL PARAMETERS
Cin      rvalue   character string
Cin      fvalue   character string
Cin      len      length of strings
C#COMMON BLOCKS
C        none
C#CALLER asndbr
C#METHOD
C     Comparison.
C##
```

```
C     ASODEL ************************************************
$CONTROL segment=ASGNO
C$TRACE asodel;
      SUBROUTINE asodel(mcyds,yard,nclas,mccls,clist)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      integer mcyds,mccls,nclas
      character*(mcyds) yard
      character*(mccls) clist(nclas)
C*                                          *** ABSTRACT ***
C#PURPOSE   ASsigner Outbound tuple DELetion utility.  Deletes
C     a tuple from ncjodat.projj if appropriate.
C#AUDIT HISTORY
C        MSCarey        03-jul-83  AUTHOR
C#FORMAL PARAMETERS
Cin      mcyds    # chars in yard name
Cin      yard     name of yard now being processed
Cin      nclas    number of classes in this yard
Cin      mccls    number of characters in a class name
Cin      clist    list of classes in this yard
C#COMMON BLOCKS
Cin      asoprm   outbound parameters
Cin      asocrs   outbound cursors
Cio      astfr    tuple buffers
Cin      asnvld   valid yards classes job types
Cin      fld05    field list for ncjodat.projj
C#CALLER asndbr
C#METHOD
C     Find out if the current tuple is one of interest for this
C     invocation.  If not, return.  If so, delete it unless it is
C     flagged and get the next tuple.  If it is flagged, then see
C     if it is one that was already processed by ashard.  That will
C     be true if its (yard class jobt) is the same as that of the
C     PREVIOUS tupfil record, indicating that this tuple is a trailer
C     in the ordering sequence but is still of a type processed by
C     ashard (since there is a tuple in ashard with the same key).
C##
```

```
C     ASPFLD ************************************************
$CONTROL segment=ASGNO
C$TRACE aspfld;
      SUBROUTINE aspfld(yard,class,all)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      character*8 yard,class*12
      logical all
C*                                              *** ABSTRACT ***
C#PURPOSE   ASsigner Planning Factor LoaDer.  Reads one or
C     all job description tuples for a given yard-class-job type
c     from ncjdat.descj and stores them in /asjd/.
C#AUDIT HISTORY
C        MSCarey        27-jun-83  AUTHOR
C#FORMAL PARAMETERS
Cin      yard     name of yard to find job desc for
Cin      class    class and job type to find desc for
Cin      howmny   true if caller wants all tuples matching
C                 input key, false if only first one needed
C#COMMON BLOCKS
Cin      scenar   scenario field value information
Cin      asoprm   outbound parameters
Cout     asjd     job description holding buffer
Cout     rcrd06   interface buffer for relation ncjdat.descj
C#CALLER various assigner outbound
C#METHOD
C     Convert the passed key values to the proper form for DB query.
C     Point to the first match and place in the holding area.
C     Read the rest of the matches if desired.
C##
```

```
C     ASPFTR **************************************************
$CONTROL segment=ASGNO
C$TRACE aspftr:
      SUBROUTINE aspftr(recnum)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      integer recnum
C*                                                *** ABSTRACT ***
C#PURPOSE   ASsigner Planning Factor TRansfer.  Moves a job
C     description record from rcrd06 to row recnum of /asjd/
C#AUDIT HISTORY
C        MSCarey        29-jun-83  AUTHOR
C#FORMAL PARAMETERS
Cin      recnum   record number (row) to move data to in asjd
C#COMMON BLOCKS
Cin      rcrd06   holds job desc record extracted from ncjdat
Cio      asjd     holds many job description records
C#CALLER aspfld
C#METHOD
C     Lots of assignments
C##
```

```
C     ASPRED*************************************************
$CONTROL segment=ASGNO
      SUBROUTINE aspred(nclash)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      integer nclash
C*                                           *** ABSTRACT ***
C#PURPOSE   Spreads out schedule dates reasonably evenly.
C#AUDIT HISTORY
C        MSCarey        11-aug-83  AUTHOR
C#FORMAL PARAMETERS
Cin      nclash   number of classes in record buffer
C#COMMON BLOCKS
Cin      asoprm   assigner outbound parameters
Cio      asrbuf   buffer holding ships/dates to spread
Cin      lprnts   debug switches
C#CALLER asncnv
C#METHOD
C     The work area here is /asrbuf/, which is in order of
C     adjust or spreading date of the ships in the current
C     complexity group.  There is also a linked-list ordering
C     by class.
C
C     Figure out the first and last allowable adjdat-s, and
C     from these and the number of ships get the size of the
C     interval with which to spread all the ships evenly. Do
C     so.
C
C     Then go through each class, looking for dates earlier or
C     later than their limits.  On finding an earlier, add the
C     minimum number of days to bring it up to its lower limit, and
C     add this number to all subsequent ships of the class.
C     On finding one later than its limits, group all ships in the
C     class with the same limits and spread evenly within those limits,
C     doing nothing to ships in following periods.
C##
```

```
C     ASPRD2*****************************************************
$CONTROL segment=ASGNO
      SUBROUTINE asprd2(nclash)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      integer nclash
C*                                                *** ABSTRACT ***
C#PURPOSE   Spreads out schedule dates reasonably evenly.
C          This version is original, now is secondary.
C#AUDIT HISTORY
C        MSCarey        11-aug-83  AUTHOR
C#FORMAL PARAMETERS
Cin      nclash   number of classes in record buffer
C#COMMON BLOCKS
Cin      asoprm   assigner outbound parameters
Cio      asrbuf   buffer holding ships/dates to spread
Cin      lprnts   debug switches
C#CALLER asncnv
C#METHOD
C     The work area here is /asrbuf/, which is in order of
C     adjust or spreading date of the ships in the current
C     complexity group.  There is also a linked-list ordering
C     by class.
C
C     Figure out the first and last allowable adjdat-s, and
C     from these and the number of ships get the size of the
C     interval with which to spread all the ships evenly. Do
C     so.
C
C     Then go through each class, looking for dates earlier or
C     later than their limits.  On finding an earlier, add the
C     minimum number of days to bring it up to its lower limit, and
C     add this number to all subsequent ships of the class.
C     On finding one later than its limits, group all ships in the
C     class with the same limits and spread evenly within those limits,
C     doing nothing to ships in following periods.
C##
```

```
C     ASTUPF ***********************************************
$CONTROL segment=ASGNO
C$TRACE asycls;
      SUBROUTINE astupf(mccls,nclash,hldcls,mcyds,yard,timyd,orgpos,
     1                  hldnum,nclas,tupfst,nxtupf)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      integer mccls,nclash,mcyds,nclas,nxtupf,tupfst(nclas)
      integer orgpos(nclas),hldnum(nclas)
      integer*4 timyd
      character*(mccls) hldcls(nclash), yard*(mcyds)
C*                                        *** ABSTRACT ***
C#PURPOSE   Converts /asrbuf/ records to tupfil tuple images.
C     When all classes in yard have been processed, these tuple
C     images will be used to update the data base.
C#AUDIT HISTORY
C        MSCarey          20-jun-83  AUTHOR
C#FORMAL PARAMETERS
Cin      hldcls   name of each class
Cin      ydname   name of yard holding these classes
Cin      timyd    time stamp for this yard; used to get asnorder
Cin      orgpos   position of clist i on assigner list (for asnorder)
Cin      hldnum   position of hldcls i on clist
Cin      nclas    number of classes in this yard
Cio      tupfst   pointer to first tuple image for class i
Cio      nxtupf   next free record in tupfil
C#COMMON BLOCKS
Cin      asrbuf   holds 1-ship 1-record structure for this comp-grp
Cio      asjd     job descriptions
Cout     astfr    record for reading/writing tupfil
C#CALLER asncnv
C#METHOD
C     Loop over the number of ships in each class.  Construct a tuple
C     image for each of these ships (load planning factors for class
C     at the outset).  Then add this image to tupfil, maintaining
C     the pointer structure.
C##
```

```
C     ASUBDL ***********************************************
$CONTROL segment=asgno
      SUBROUTINE asubdl(clchr,class,hull,comnum,map)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      integer hull,comnum,clchr
      integer*4 map
      character*(clchr) class
C*                                         *** ABSTRACT ***
C#PURPOSE   Deletes tuples in relations subsidiary to
C     schedule relation (ncjodat.projj).  Hardwires only.
C#AUDIT HISTORY
C        MSCarey        09-aug-83  AUTHOR
C#FORMAL PARAMETERS
Cin      clchr    chars in class
Cin      class    name of class to delete
Cin      hull     hull number of ship to delete
Cin      comnum   commissioning number of ship to delete
Cin      map      bit mapped variable indicating which relations
C                 to delete tuples from.  Mapping is: [0 --> 15]
C                 15:ncjodol  14:ncjolbr  13:ncjoemp  12:ncjomr
C                 11:ncjomd   10:ncjocom
C#COMMON BLOCKS
Cin      scenar   holds current scenario information
C#CALLER ashard,asodel
C#METHOD
C     Set up the point transfer buffer.
C     Parse the bit map to see which relations to look in dynamically.
C     If a bit is on, make sure that its relation is open.
C     Point towards the target record, then delete it.
C     Look for trailers with the same key value.
C##
```

```
C     ASYCLS ***********************************************
$CONTROL segment=ASGNO
C$TRACE asycls;
      SUBROUTINE asycls(iyard,lstchr,lstlen,clist,cptr,orgpos,nclas)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      integer iyard,lstchr,lstlen
      integer nclas,cptr(lstlen),orgpos(lstlen)
      character*(lstchr) clist(lstlen)
C*                                           *** ABSTRACT ***
C#PURPOSE   ASigner Yard Classes.  Constructs a sorted list
C           of the non-repair job classes in yard iyard.
C#AUDIT HISTORY
C        MSCarey         19-jun-83  AUTHOR
C#FORMAL PARAMETERS
Cin      iyard    index of the yard in /casgn/
Cin      lstchr   max length of any class name
Cin      lstlen   max number of classes to be returned
Cout     clist    the sorted list of classes (job type char attached)
Cout     cptr     bufasn record pointers for each class
Cout     orgpos   original position in display order of clist(i)
Cout     nclas    number of classes returned
C#COMMON BLOCKS
Cio      asgn     bufasn and editing-phase blocks
C#CALLER asncnv
C#METHOD
C     Do gets from bufasn until no more in iyard.  Check for
C     repair job types and ignore them.  Retain the pointer
C     for each class gotten.  Sort the final list and pointers
C##
```

```
C     CKPF ************************************************
$CONTROL segment=asgnd
      SUBROUTINE ckpf(clsjob,yard,holdup,ok)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      character clsjob*12,yard*8
      logical holdup,ok
C*                                           *** ABSTRACT ***
C#PURPOSE   Checks to make sure there are planning factors for
C     new class-jobs entered by user.
C#AUDIT HISTORY
C        MSCarey        29-mar-84  AUTHOR
C#FORMAL PARAMETERS
Cin      clsjob   name of ship class as entered, with job type char
Cin      yard     name of yard job assigned to
Cin      holdup   true if should stop execution to ensure msg seen
Cout     ok       true if job desc found
C#COMMON BLOCKS
Cin      asnvld   validity info
C#CALLER asnadd
C#METHOD
C     Decode clsjob into data base key values and do a point
C     on the job description cursor; success of point means ok.
C#LOCAL VARIABLES
C        jchar    job type, single-character representation
C        target   dummy data destination
C##
```

```
C     CMNGET******cmnsav******scnget************************
$CONTROL segment=asgni,check=3
      SUBROUTINE cmnget
      CHARACTER*16 scn
C*                                          *** ABSTRACT ***
C#PURPOSE Get-Save /casgn/ & /nasgn/ in asgn include - unit uasnc
C#AUDIT HISTORY
C         Densmore        04-Apr-83  AUTHOR
C#TYPE    Manual Assigner Utility
C#FORMAL PARAMETERS
Cout      scn     scenario name in disk buffer
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  asnini, asncmd
C#METHOD
C  Simply uses variable format read/write
C#LOCAL VARIABLES
C##
```

```
C     DBASIS***********************************************
$CONTROL segment=asgni,check=3
      FUNCTION dbasis(disbas,mode)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      PARAMETER mdb=8
      CHARACTER dbasis*mdb
      CHARACTER disbas*mdb
      CHARACTER mode*6
C*                                           *** ABSTRACT ***
C#PURPOSE Returns appropriate date fieldname for the job
C         relations given the menu display basis string and
C         a character mode describing the job relation
C#AUDIT HISTORY
C         Densmore        23-Jun-83  AUTHOR
C#TYPE    assigner routine
C#FORMAL PARAMETERS
Cin       disbas  display basis, from menu system; describes
C                 the date to be used in determining the period
C                 to which a job (tuple) belongs; one of:
C                 'AWARD','START','KEEL','LAUNCH','DELIVERY'
Cin       mode    type of job relation; one of:
C                 'NEWCON','REPAIR'
C#CALLER  asntpx,asntpi
C#METHOD  trivial; if mdb is changed from 8 note required code changes
C##
```

```
C     GETASN******putasn**************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE getasn(ptr)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER ptr
C*                                         *** ABSTRACT ***
C#PURPOSE Gets/Puts assignment record into position ptr
C#AUDIT HISTORY
C          Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       ptr     assignment record position
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  various ASN routines
C##
```

```
C     INICLS*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE inicls(class,order,loc)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
C *** CHARACTER*mccls class
      INTEGER*4 order
      INTEGER loc
C*                                           *** ABSTRACT ***
C#PURPOSE Sets up sums and increments for a new class buffer
C#AUDIT HISTORY
C          Densmore        10-May-83  AUTHOR
C#TYPE    assigner routine
C#FORMAL PARAMETERS
Cin       class   class name for new buffer
Cin       order   ship order (asn order)
Cin       loc     yard location in which new buffer resides
C#COMMON BLOCKS
Cin       asgn    assigner data block
C#CALLER  asnadd,asnins
C#METHOD
C  straightforward
C##
```

```
C     INIYRD*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE iniyrd(loc,yardnm)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
C *** CHARACTER*mcyds yardnm
      INTEGER loc
C*                                           *** ABSTRACT ***
C#PURPOSE Finishes the process of adding a yard
C#AUDIT HISTORY
C         Densmore        10-May-83  AUTHOR
C#TYPE    assigner routine
C#FORMAL PARAMETERS
Cin       loc     yard index
Cin       yardnm  yard name
C#COMMON BLOCKS
Cout      asgn    assigner data block
C#CALLER  newyrd
C#METHOD
C  straightforward
C##
```

```
C     LOCYRD*****************************************************
$CONTROL segment=asgnd,check=3
      INTEGER FUNCTION locyrd(yard,names,len)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER len
C *** CHARACTER*12 yard,names(len)  -- 12==mcyds /asgn/
C*                                            *** ABSTRACT ***
C#PURPOSE Locates yard in names array via binary search
C#AUDIT HISTORY
C         Densmore        17-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       yard    yard name to locate
Cin       names   sorted array of yard names to search
Cin       len     length of names
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  asnadd
C#METHOD
C  Binary search.  If the yard is not found, locyrd is returned
C  such that if the yard were inserted it would become number
C  locyrd and items locyrd on would be right-shifted.  NOTE: locyrd
C  assumes the dimension of NAMES to be at least LEN+1 on failure.
C#LOCAL VARIABLES
C         left,right -- search positions
C         locyrd, the returned value, is also used as the mid value
C##
```

```
C     NEWYRD*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE newyrd(loc,defind)
C*                          *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER loc
      LOGICAL defind
C*                                               *** ABSTRACT ***
C#PURPOSE adds new yard
C#AUDIT HISTORY
C          Densmore         16-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
ClocType  loc     yard index; returned 0 if abort is desired
Cin       defind  (IF defind THEN locType="in/out" ELSE locType="out")
C#COMMON BLOCKS
Cin/out   asgn    assigner data
Cin       scrchr  screen characters
C#CALLER  asnadd
C#METHOD
C  Checks for input and system errors; obtains yard name input;
C  Right shifts appropriate arrays to keep these arrays sorted
C  by yard name.
C#LOCAL VARIABLES
C         yardnm  local yard name input
C         lenr    length of right shift
C         msg     character buffer for asnpro message
C         leny    length of yard name
C##
```

```
C     NXTCLS************************************************
$CONTROL segment=asgni,check=3
      SUBROUTINE nxtcls(cursor,fields,vldydi,firstd,lastd,
     1 tupmax,tuple,len)
C*                      *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER cursor,vldydi,len
      CHARACTER fields*255
      INTEGER tupmax(len),tuple(len)
      INTEGER*4 firstd,lastd
C*                                          *** ABSTRACT ***
C#PURPOSE Grabs the next tuple from the relation given by cursor
C#AUDIT HISTORY
C         Densmore        22-Jun-83  AUTHOR
C#TYPE    assigner inbound routine
C#FORMAL PARAMETERS
Cin       cursor  cursor index to appropriate relation
Cin       fields  DTS string giving names of fields to be returned
Cin/out   vldydi  index to the list of valid yards in /asnvld/
Cin       firstd  RELATE representation of first date of first period
Cin       lastd   RELATE rep of last date of last period
Cin       tupmax  maximum value of a tuple...used when vldydi max
C                 value is exceeded to set 'tuple', so that the
C                 hash sorting done by the caller still works
Cout      tuple   returned tuple value
Cin       len     length of tuple...better be .GE. than fields implies
C#COMMON BLOCKS
Cin       asnvld  assigner valid lists
C#CALLER  asntpx,asntpx$asntpi
C#METHOD
C        Nxtcls is divided into two parts: a yard search section
C  and a class search section.  The routine performs an initial-
C  ization part first if vldydi is zero on entry.  Note that
C  NeXT-Tuple operations are attempted before CALc operations are
C  performed during searches for a next valid yard.  Next ops are
C  much less expensive than are Calc ops.
C#LOCAL VARIABLES
C         eot     True when no more tuples in that relation
C                 or when a calc operation failed
C         yard,class,series,       -  tuple elements unloaded
C         scnaro,type,date,order  /
C         clcomp  *mcvcls version of class, for comparison purposes
C##
```

```
C     NXTCLZ**************************************************
$CONTROL segment=asgni,check=3
      SUBROUTINE nxtclz(routin,where,tuple,string,ls,eot)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER ls,tuple(1)
      CHARACTER routin*6,where*255,string*(ls)
      LOGICAL eot
C*                                                *** ABSTRACT ***
C#PURPOSE Prints diagnostics for NXTCLS
C#AUDIT HISTORY
C         Densmore        04-Jul-83  AUTHOR
C#TYPE    assigner inbound diagnostic routine
C#FORMAL PARAMETERS
Cin       routin  *6 caller name
Cin       where   DTS description of where the call is being made
Cin       tuple   integer version of tuple read, if any
Cin       string  calc string, or whatever else, of length ls
Cin       ls      length of string
Cin       eot     True if no tuple was read because end-of-...
C#COMMON BLOCKS
Cin       lprnts  diagnostic common block
C#CALLER  nxtcls
C##
```

```
C     REMCLS*****************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE remcls(loc,before,item,after)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER loc,before,item,after
C*                                         *** ABSTRACT ***
C#PURPOSE Performs decrements and repointering for removal of a class
C#AUDIT HISTORY
C         Densmore        06-Jun-83  AUTHOR
C#TYPE    assigner routine
C#FORMAL PARAMETERS
Cin       loc     yard index
Cin       before  pointer to class before the one to be deleted
Cin       item    pointer to the one scheduled for deletion
Cin       after   pointer to the next one after item
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  asnval,asndel
C#METHOD
C  Decrements asntot,numasn,sumper,grdtot.  Re-route pointers.
C##
```

```
C     REMYRD*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE remyrd(loc)
C*                       *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER loc
C*                                          *** ABSTRACT ***
C#PURPOSE Removes an entire yard
C#AUDIT HISTORY
C         Densmore        16-Mar-83  AUTHOR
C#TYPE    manual assigner routine
C#FORMAL PARAMETERS
Cin       loc     yard index location
C#COMMON BLOCKS
Cin       scrchr  screen characters
Cin/out   asgn    assigner data
C#CALLER  asndel
C#METHOD
C  Prompts to make sure if Prompt=.True.; then runs down the
C  assignment buffer list and decrements all summary arrays.
C  The list is then CONSed to the free chain, and all approp-
C  riate arrays are left-shifted.
C#LOCAL VARIABLES
C         msg     asnpro message buffer
C         verify  input containing "^" or "?"
C         item    assignments buffer pointer
C         lenl    length of left shift
C##
```

```
C     SVSTAT******restat*************************************
$CONTROL segment=asgni,check=3
      SUBROUTINE svstat
C*                                                   *** ABSTRACT ***
C#PURPOSE save/restore screen status for line printer outputs
C#AUDIT HISTORY
C         Densmore        11-May-83  AUTHOR
C#TYPE    assigner utility
C#CALLER  asnlpr
C##
```

```
C     TMSTMP ***********************************************
$CONTROL segment=ASGNO
      INTEGER*4 FUNCTION tmstmp(idum)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
      integer idum
C*                                       *** ABSTRACT ***
C#PURPOSE   Returns current time in seconds since 1/1/1983
C#AUDIT HISTORY
C        MSCarey        03-jul-83  AUTHOR
C#FORMAL PARAMETERS
Cin      idum      dummy
C#COMMON BLOCKS
Cin      tddate    date manipulation functions
C#CALLER asncnv
C#METHOD
C     Get the current date and convert it to seconds; then get
C     the time of day and add it on.
C##
```

```
C     TUPFRD ***********************************************
$CONTROL segment=ASGNO
      SUBROUTINE tupfrd(recnum,iclas,tupfst,nclas,eofil,
     1                  lstyrd,lstcls,lstjob)
C*                        *** FORMAL PARAMETER DECLARATIONS ***
      integer recnum,iclas,nclas,tupfst(nclas)
      logical eofil
      character lstyrd*8,lstcls*10,lstjob*6
C*                                              *** ABSTRACT ***
C#PURPOSE   Read a record from direct access tuple holding file.
C#AUDIT HISTORY
C        MSCarey        01-jul-83  AUTHOR
C#FORMAL PARAMETERS
Cio      recnum   in: record to read; out: next record
cio      iclas    location on clist of current class-job type
Cin      tupfst   pointers to first tupfil record each class-job
Cin      nclas    number of class-job types this yard
Cio      eofil    true if no more tupfil records
Cout     lstyrd   yard for tupfil record in memory on call
Cout     lstcls   class "
Cout     lstjob   job type "
C#COMMON BLOCKS
Cin      asoprm   outbound parameters
Cout     astfr    tuple/record holding records
C#CALLER asdbr
C#METHOD
C     Read and reset recnum
C##
```

```
C     VALCLS***********************************************
$CONTROL segment=asgnd,check=3
      LOGICAL FUNCTION valcls(class)
C*                    *** FORMAL PARAMETER DECLARATIONS ***
C *** CHARACTER*mccls class
C*                                          *** ABSTRACT ***
C#PURPOSE Determine if input class is allowed in this scenario
C#AUDIT HISTORY
C        Densmore        02-Jun-83  AUTHOR
C#TYPE    assigner routine
C#FORMAL PARAMETERS
Cin       class   input class name...character*mccls
C#COMMON BLOCKS
Cin       asgn    assigner data block
C#CALLER  asnadd
C#METHOD
C  Matches against legal list; if match then verifies validity
C##
```

```
C     VALYRD*****************************************************
$CONTROL segment=asgnd,check=3
      LOGICAL FUNCTION valyrd(yard)
C*                     *** FORMAL PARAMETER DECLARATIONS ***
C *** CHARACTER*mcyds yard
C*                                            *** ABSTRACT ***
C#PURPOSE Determines if input yard is valid in this scenario
C#AUDIT HISTORY
C         Densmore        02-Jun-83  AUTHOR
C#TYPE    assigner routine
C#FORMAL PARAMETERS
Cin       yard    input yard
C#COMMON BLOCKS
Cin       asgn    assigner data block
C#CALLER  newyrd
C#METHOD
C  Verifies that input yard is on the valid yards list
C##
```

```
C     VLDLST*************************************************
$CONTROL segment=asgni,check=3
      SUBROUTINE vldlst
C*                                              *** ABSTRACT ***
C#PURPOSE Initializes valid lists for assigner (classes,yards)
C#AUDIT HISTORY
C         Densmore       10-Jun-83  AUTHOR
C#TYPE    assigner routine
C#COMMON BLOCKS
Cin       scenar  current scenario
Cout      asnvld  valid lists
C#CALLER  asnini
C#METHOD
C  simply uses liston for each list being initialized
C##
```

```
C     VLDLSZ*************************************************
$CONTROL segment=asgni,check=3
      SUBROUTINE vldlsz
C*                                           *** ABSTRACT ***
C#PURPOSE Prints diagnostic information for vldlst on /asnvld/
C#AUDIT HISTORY
C         Densmore        05-Jul-83  AUTHOR
C#TYPE    assigner inbound diagnostic
C#COMMON BLOCKS
Cin       asnvld  assigner valid arrays
C#CALLER  vldlst
C#METHOD
C  just a set of prints and trecol calls
C##
```

```
C     YCASN***************************************************
$CONTROL segment=asgni,check=3
      SUBROUTINE ycasn
      LOGICAL flag
C*                                                   *** ABSTRACT ***
C#PURPOSE Implements Control-Y in Refresh
C#AUDIT HISTORY
C         Densmore         06-Apr-83  AUTHOR
C#TYPE    Assigner utility
C#METHOD
C  To use this routine with a specified module, one relies on an
C  unspecified compiler specific mechanism to call YCASN on some
C  user requested interrupt.  For example, on the HP, the exec-
C  ution of the Fortran statement "   ON CONTROLY CALL YCASN"
C  performed this function.  This routine assumes that following
C  completion of the interrupt process (ie. the call to YCASN)
C  control returns to where it was before the interrupt.  Thus,
C  to discover that the interrupt occurred, entry YCASNR may be
C  called; it returns .TRUE. if so, and .FALSE. otherwise, while
C  resetting the internal flag (save).  YCASNI may be called to
C  initialize this process.  Note that the ON statement need not
C  be reset, since calling YCASN is harmless, and only effective
C  if YCASNR is being called in a loop or something like that.
C##
```

```
C     YRDCPY*************************************************
$CONTROL segment=asgnd,check=3
      SUBROUTINE yrdcpy(from,loc,defind,succes)
C*                         *** FORMAL PARAMETER DECLARATIONS ***
      INTEGER from,loc
      LOGICAL defind,succes
C*                                               *** ABSTRACT ***
C#PURPOSE Implements yard copy
C#AUDIT HISTORY
C         Densmore        08-Jul-83  AUTHOR
C#TYPE    assigner routine
C#FORMAL PARAMETERS
Cin       from    from yard
Cin       loc     to yard
Cin       defind  .TRUE. if loc is defined
Cout .    succes  true if successful
C#COMMON BLOCKS
Cin/out   asgn    assigner data block
C#CALLER  asncpy
C#METHOD
C       This copy-yard part does the customary checks and then adds
C  the new yard via a newyrd call.  Then it is assured that enough
C  class buffers are actually allocated on the free chain to permit
C  using it without changing any of the pointers within it.  Finally,
C  the appropriate data is placed in each class buffer by looping
C  over the from buffers.
C#LOCAL VARIABLES
C         ochain  pointer to current part of old yard class chain
C         nchain  pointer for new yard class chain
C         from    old or from yard location
C         loc     new or to yard location
C         new     used as a do index
C         item    pointer to current buffer
C         end     pointer to last buffer in allocated chain
C##
```

APPENDIX A

CROSS REFERENCE FROM AUTOMATED DATA SYSTEMS DOCUMENTATION STANDARDS CONTENTS TO ALIAS GUIDES CONTENTS

A.0 PURPOSE OF THE APPENDIX

The set of manuals which form the documentation for ALIAS do not conform strictly to the DoD 7935.1-S documentation standard. They contain all the information mandated by the standard (with the exception of Functional Description, Test Plan, and Test Report) and more, but are organized differently. The organization of the standard is not well suited to ALIAS, and would have resulted in much less useful documentation.

This Appendix lists sections in the ALIAS Guides which contain the information mandated in each section of the standard. It is organized according to the tables of contents for the standard manuals, with references to one or more sections in the documentation as written. A reader wishing to have information presented in the order given by the standard tables of contents may detach this Appendix and use it to order his reading of the documentation.

In a few cases, sections mandated by the standard were not relevant to ALIAS. Comments regarding this are included in this Appendix.

In order to conserve space, references to the various guides are made according to the following scheme: a section in a particular guide is designated as G-#[.#.#...], where G represents the code for the guide and #.#.... is the actual section number within it. Codes for the Guides are:

U:	Alias User's Guide
P:	Alias Guide to System Maintenance and Expansion
D:	Alias Data Base Reference Guide
X:	Any ALIAS Guide

A.1 SYSTEM/SUBSYSTEM SPECIFICATION

1.0 GENERAL

1.1 Purpose of the System/Subsystem Specification
see mainly P-1.1

1.2 Project References
see P-1.1.3, P-1.2, P-1.3

1.3 Terms and Abbreviations
see U-2, P-2

2.0 SUMMARY OF REQUIREMENTS

2.1 System/Subsystem Description
see U-1.3, P-1.2, P-1.3

2.2 System/Subsystem Functions
see P-1.2, P-1.3, P-8, P-11 and onward

2.2.1 Accuracy and Validity
no reference

All ALIAS calculations must be carried out with a normal degree of accuracy; that is, the nature of the problems are not such that extraordinary mathematical precision is required, as it sometimes is for scientific problems.

2.2.2 Timing
see P-2.3.11, P-1.3.2.7

In general, response time should be minimized, and for functions requiring a great deal of time, off-line execution options should be available.

2.3 Flexibility
see P-1.3, P-1.4, P-2, P-6, P-8, P-9.1

3.0 ENVIRONMENT
see P-6

3.1 Equipment Environment
see P-4

3.2 Support Software Environment
see P-4, P-5

3.3 Interfaces
see P-1.3.2.5, P-2.3.8, P-2.3.9, P-2.2.5, P-2.3.2, P-8.2.5, P-8.3.5, P-8.4.5, P-9, P-10, P-11 and onward

3.4 Security and Privacy
see P-7, P-8.3, P-8.4, P-11 and onward

3.5 Controls
see P-7, P-8

4.0 DESIGN DETAILS

4.1 General Operating Procedures
see P-1.3, U-1.3, U-4, U-5

4.2 System Logical Flow
see P-1.3, P-3.2, P-8.1, P-11 and onward

4.3 System Data
see U-5, P-3.2, P-8.2.4 and onward

4.4 Program Descriptions
see P-8, P-10, P-11 and onward

A.2 <u>PROGRAM SPECIFICATIONS</u>

1.0 GENERAL

1.1 Purpose of the Program Specification
see P-1.1, P-8, P-11 and onward

1.2 Project References
see P-1.1.3, P-1.2, P-1.3

1.3 Terms and Abbreviations
see U-2, P-2

2.0 SUMMARY OF REQUIREMENTS
see P-1.2, P-2, P-8.2.1, P-8.3.1, P-8.4.1, P-11 and onward

General system requirements and standards are covered in the early sections of the Maintenance Guide, while specifics for each system module are covered in Section 8 and Sections 11-12. The remarks on accuracy and validity made above (A.2-2.2.1) apply.

3.0 ENVIRONMENT
see P-6

3.1 Equipment Environment
see P-4

3.2 Support Software Environment
see P-4, P-5

3.3 Interfaces
see P-1.3.2.5, P-2.3.8, P-2.3.9, P-2.2.5, P-2.3.2, P-8, P-9, P-10, P-11 and onward

3.4 Security and Privacy
see P-7, P-8.3, P-8.4, P-11 and onward

3.5 Controls
see P-7

4.0 DESIGN DETAILS
see P-8, P-11 and onward

A.3 <u>DATA BASE SPECIFICATIONS</u>

1.0 GENERAL

1.1 Purpose of the Data Base Specification
see D-1.1

1.2 Project References
see P-1.1.3, P-1.2, P-1.3, I-1

1.3 Terms and Abbreviations
see U-2, P-2

2.0 DATA BASE IDENTIFICATION AND DESCRIPTION

2.1 Data Base Identification
see D-1.4

2.1.1 System Using the Data Base
see D-1.3

2.1.2 Effective Dates
no reference
The ALIAS data base may be used with the ALIAS system for as long as the system is in existence. The data base is expected to expand and change continuously.

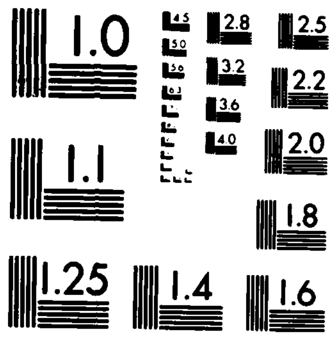
AD-A150 423 ALIAS (ACQUISITION AND LOGISTICS INFORMATION AND ANALYSIS SYSTEM) MAINTEN. (U) DECISION-SCIENCE APPLICATIONS INC ARLINGTON VA M S CAREY ET AL. 7/7

UNCLASSIFIED 31 OCT 84 DSA-593-VOL-2 N00014-82-C-0813 F/G 15/5 NL

END

FILMED

DTIC


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

2.1.3 Storage Requirements
see D-2

2.1.4 Physical Description of Data Base Files
see D-2

2.2 Labeling/Tagging Conventions
see D-2

2.3 Organization of the Data Base
see D-2, D-5

2.4 Special Instructions
see D-3, D-4, D-5, D-6

2.5 Support Programs Available for Handling the Data Base
see D-4, D-6

3.0 DATA DEFINITIONS

3.1 Data Files
see D-2

3.2 Tables
same as files

3.3 Items
see D-2

3.4 Records and Entries
not applicable

4.0 INTEGRATED DATA BASE
See D-1.4, D-2.0, D-5, D-6

A.4 <u>USERS MANUAL</u>

1.0 GENERAL

1.1 Purpose of the Users Manual
see mainly U-1.1

1.2 Project References
see P-1.1.3, P-1.2, P-1.3

1.3 Terms and Abbreviations
see U-2, P-2

1.4 Security and Privacy
see U-7

2.0 SYSTEM SUMMARY

2.1 System Application
see U-1.2, U-1.3, U-2.0

2.2 System Operation
see U

2.3 System Configuration
see U-4

2.4 System Organization
see U-1.2, U-1.3, U-5

2.5 Performance
see U-1.2, U-1.3, U-4, U-5, U-7 and onward

2.6 Data Base
see U-6

2.7 General Description of Inputs, Processing, and Outputs
see U-1.3, U-5, U-7 and onward

3.0 STAFF FUNCTIONS RELATED TO TECHNICAL OPERATIONS

3.1 Initiation Procedures
see U-1.3, U-1.4, U-3, U-4

3.2 Staff Input Requirements
see U-1.4, U-3 and onward

3.3 Output Requirements
see U, P-8 and onward

4.0 FILE QUERY PROCEDURES

4.1 System Query Capabilities
see U-1.4, U-6, U-7, RELATE manuals

4.2 Data Base Format
see D-2

4.3 Query Preparation
see U-7, RELATE manuals

4.4 Control Instructions
not relevant

A.5 COMPUTER OPERATION MANUAL

Not provided for; structure of system does not call for separate class of operators. Program inventories, file inventories, processing and security descriptions may be found in the User's and Maintenance Guides.

A.6 PROGRAM MAINTENANCE MANUAL

1.0 GENERAL

1.1 Purpose of the Program Maintenance Manual
see mainly P-1.1

1.2 Project References
see P-1.1.3, P-1.2, P-1.3, I-1

1.3 Terms and Abbreviations
see U-2, P-2

2.0 SYSTEM DESCRIPTION

2.1 System Application
see P-1.2, P-1.3

2.2 Security and Privacy
see P-7

2.3 Program Description
see P-8, P-3, P-6, P-10, P-11 and onward

3.0 ENVIRONMENT

3.1 Equipment Environment
see P-4

3.2 Support Software
see P-5

3.3 Data Base
see D

4.0 PROGRAM MAINTENANCE PROCEDURES

4.1 Conventions
see P-2, P-6

4.2 Verification Procedures
see P-9

4.3 Error Conditions
see U-B

4.4 Special Maintenance Procedures
see P-6

4.5 Special Maintenance Programs
see P-6

4.6 Listings
see P-Appendix C

APPENDIX B

SUMMARY OF ALIAS HOST SYSTEM DEPENDENCIES

This Appendix is meant as an introduction to the tasks which would need to be completed in order to convert ALIAS to run on a host computer other than an HP-3000. ALIAS is fundamentally a very host-dependent system, primarily because of its dependence on RELATE and BUILDER, but measures were taken during development to isolate dependency in order to minimize conversion costs.

This Appendix should not be construed as a complete listing of conversion requirements. Such a listing inevitably depends on the hardware and software of the target host as well as on the existing software.

The major host dependencies fall into four categories: RELATE, BUILDER, HP FORTRAN, and MPE dependencies.

B.1 RELATE DEPENDENCE

RELATE is the DBMS used to implement and access the ALIAS data base. A move to a new system will involve creation of the ALIAS data base structure using the DBMS on the new host, an unload of the data contained in RELATE files on the HP 3000, and a reload on the new host.

Although tedious and time consuming, these tasks are unlikely to present serious technical challenges or suprises. Converting ALIAS FORTRAN programs to access the new data base programmatically may be another matter. Such programmatic access must go through the equivalent of the RELATE Host Language Interface. Should the new DBMS fail to provide a cursor-oriented, routine-call oriented interface, major revision of all ALIAS FORTRAN code would be required.

Given a cursor-oriented, general-purpose routine call interface structurally similar to the RELATE HLI (really the most common interface method among DBMSs) there may not be much problem. Although calling syntax and data structures may differ, (almost) all ALIAS programmatic DBMS usage is buffered through the DBIF, a library of interface routines written specifically to support future conversions. These are general-purpose routines whose formal parameters could remain unchanged while their guts were rewritten to work with the new DBMS. Should this be the case conversion changes would be isolated in perhaps two dozen routines and a few thousand lines of code.

There are two reasons why things might not be so rosy (in addition to the disaster of a non-cursor-oriented new DBMS). First, RELATE requires that data source and target buffers be word-aligned arrays or common blocks in which numeric and character variables are mixed according to data relation field data type. Many FORTRAN '77 compilers will not permit such constructs, making it possible that a different buffering method will be required. Since these buffers are just passed through DBIF calls, any changes to the buffering scheme would affect all ALIAS FORTRAN routines which use RELATE.

Second, most ALIAS modules are heavily dependent on the existence of RELATE's record-point query capability, as implemented in the rtpcal DBIF routine, and its particular idiosyncracies. This dependency was necessitated by the large memory usage and execution time penalties imposed by trying to make queries using selections. Should the new DBMS fail to have a point capability, the design logic of many ALIAS query and update routines would have to be substantially changed. This might be desirable in any case, though, in order to take advantage of the efficiency features offerred by a new DBMS.

B.2 BUILDER DEPENDENCY

The BUILDER screen application generator (a member of the RELATE family of software) was used to implement the Data Base Updating system and Data Dictionary, both central elements of the ALIAS system. It is likely that complete rewrites of both modules will be required on conversion, since BUILDER currently runs only on the HP 3000. BUILDER was used because the only alternative was to write a similar package from scratch; this package would have had so many host dependencies given the limitations imposed by the HP 3000 as to be no better than use of BUILDER for conversion purposes.

CRI, BUILDER's vendor, is considering conversion of BUILDER to run under UNIX with a variety of DBMSs. Should this occur the outlook in this area might improve substantially.

In considering alternative screen application packages for the new host system particular care should be taken to ensure that all the FUNCTIONAL features of the DBU can be implemented. Few packages offer the range and power of BUILDER.

B.3 HP 3000 FORTRAN DEPENDENCY

The HP 3000's FORTRAN compiler is a nonstandard extension of the ANSI '66 standard compiler which offers many ANSI '77 standard-equivalent features. Those converting ALIAS FORTRAN programs to run on a new machine will find that a moderate effort will be required to correct syntax and logic to conform to the ANSI standard.

The major problem will be with data structures which mix character and numeric data types, technically forbidden under the '77 standard. This mixing was necessary on the HP 3000 in order to support RELATE use, as noted above.

A number of syntax differences can probably be dealt with by writing a specialized editor to process all the code and make the necessary changes. HP FORTRAN uses 's' rather than 'a' as the variable-length character output descriptor in FORMAT statements, permits the use of quotes as character string delimiters, does not support in-line string concatenation, and limits character strings to 255 characters in length.

The OPEN and CLOSE statements are also not supported by HP FORTRAN, but all functionally equivalent calls in ALIAS have been isolated in calls to the filopn and filcls utilities, sharply restricting the extent of the resulting problem.

B.4 MPE DEPENDENCIES

ALIAS is dependent on the host operating system to a moderate extent. Due to the process memory limitation of the HP 3000 a good deal of process handling is done in running modules. Conversion personnel may elect to retain this multiprocessing capability if supported by the new machine, or may simply link all modules into a single large program as was originally planned. There are advantages to each approach. In any case, the number of routines in which process handling is done is sharply limited.

Likewise, ALIAS uses MPE extra data segments as extended data memory storage in cases where the 64K byte per-process data memory limitation is binding. This paging can be eliminated on a truly virtual machine. Again, the number of routines involved is rather limited.

Most other calls to operating system service routines are isolated in general-purpose FORTRAN utilities, minimizing the work required to move onto a new system.